Elixirr

통계기반 데이터 활용

- 회귀코딩

2025.01.22

김자영 강사

머신러닝 개요

인공지능 ⊃ 머신러닝 ⊃ 딥러닝

인공지능
(Artificial Intelligence)

인간의 지능적인 행위를
흉내낼 수 있게 하는 기술

규칙기반 인공지능 포함
(추론, 탐색)

머신러닝
(Machine Learning)

컴퓨터가 **데이터**를 통해 **학습**하여
답을 도출하는 **규칙**을 찾아내는 학습법

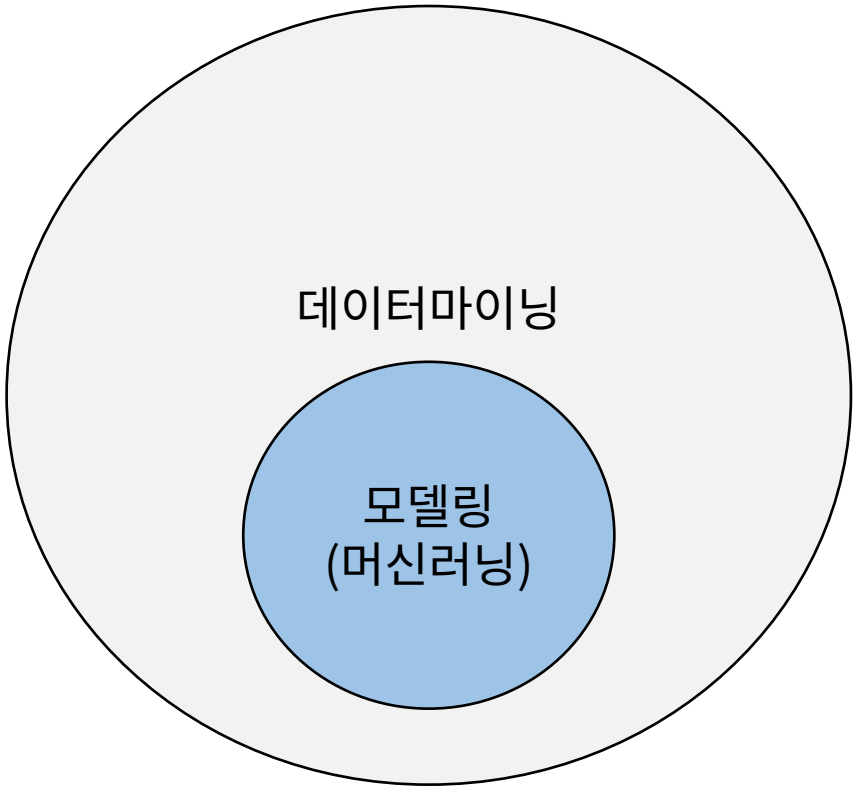
통계기반

딥러닝
(Deep Learning)

인간의 뉴런을 모방한
인공신경망 기반으로
인간과 유사하게 학습하는 방법

1950 1980 2010

데이터마이닝과 머신러닝



통찰력

데이터마이닝의 목적

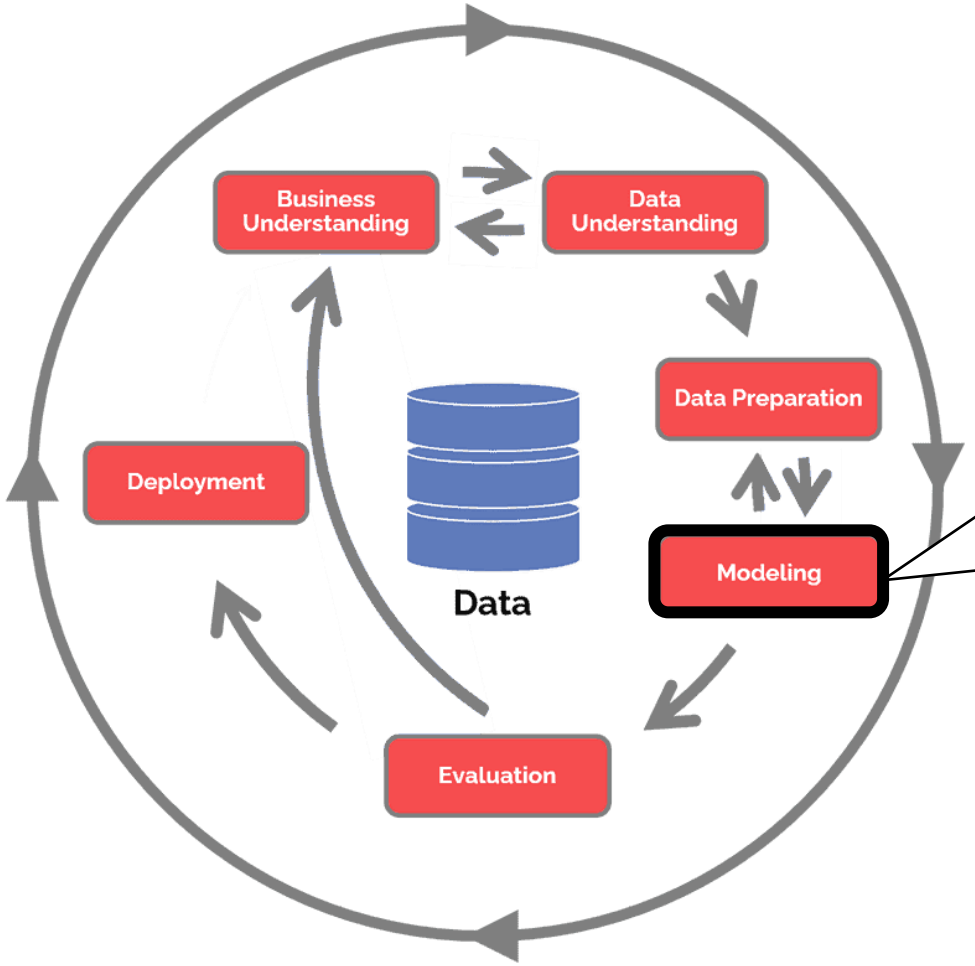
→ 데이터를 탐색하고 분석하여 **유용한 정보나 패턴을 발견**하고 이를 통해 유의미한 정보를 추출하여 의사결정에 활용한다. 데이터베이스, 통계, 데이터시각화, 머신러닝 기술 등을 사용

예측, 자동화

머신러닝의 목적

→ 주어진 데이터를 학습하여 예측을 위한 모델을 구축하고 구축된 모델을 통해 자동화된 의사결정 지원

CRISP-DM CRoss Industry Standard Process for Data Mining



그림출처 https://www.datascience-pm.com/crisp-dm-2/

- **Modeling**

  데이터에서 패턴을 학습하고

  이를 기반으로 예측이나 분류를 수행할 수 있는

  수학적 모델을 구축하는 과정

머신러닝 유형

지도학습
Supervised Learning

레이블이 있는
훈련 데이터를 사용하여
모델을 학습시키는 방법

회귀 분류

비지도학습
Unsupervised Learning

레이블이 없는
데이터에서
숨겨진 구조를 찾아내는 방법

군집화 차원축소

강화학습
Reinforcement Learning

에이전트가
환경과 상호작용하여
보상을 최대로 하도록
학습하는 방법

준지도학습
Semi-supervised Learning

레이블이 있는 데이터와
없는 데이터를 함께 사용하는 방법

머신러닝 유형

지도학습
Supervised Learning

레이블이 있는 훈련 데이터를 사용하여 모델을 학습시킨다.
→ 모델을 이용하여 새로운 데이터에 대한 **예측**을 한다.

지도학습의 목표 : 예측

회귀

연속적인 값 예측
(예) 수요 예측, 부동산 가격 예측, 에너지 사용량 예측

분류

분류 예측
(예) 이진분류 : 고객 이탈 예측, 불량제품 예측, 암 여부 예측
다중분류 : 제품 등급 분류, 고객 신용등급 분류, 리뷰 감성 분류

머신러닝 유형

비지도학습
Unsupervised Learning

레이블이 없는 데이터에서 숨겨진 구조를 찾아내는 방법

군집화

유사한 특성을 가진 데이터를 그룹화
(예) 고객 세그멘테이션, 이벤트 대상자 선정

차원축소

데이터의 중요한 특성을 유지하면서 차원을 축소
(예) 이미지, 생물 정보학등 고차원 데이터 처리 성능 향상

머신러닝 유형별 기본 알고리즘

지도학습		비지도학습	
회귀	분류	군집화	차원축소
• Linear Regression • Ridge Regression • Lasso Regression	• Logistic Regression	• K-Means clustering • DBSCAN	• PCA
• Decision Trees Regression	• Decision Trees Classifier		
• Support Vector Regression			
• K-Nearest Neighbors Regression	• K-Nearest Neighbors Classifier		
• Random Forests Regression	• Random Forests Classifier		

통계기반 머신러닝을 위한 파이썬 라이브러리

전통적인 머신러닝

배열처리 및 수치연산

데이터 전처리/분석

데이터 시각화

데이터 시각화

머신러닝 프로세스

EDA → 모형 선정 → 데이터 준비 → 데이터 전처리 → 모델학습 → 모델 평가

분석 모형 선정

- 회귀
- 분류
- 군집화

훈련에 사용할 후보 변수 선택

파생변수 생성

데이터 보강

머신러닝에 적용할 수 있는 형태로 데이터 가공

- 데이터 정제 (이상치/결측치 처리)
- 데이터 변환 (스케일링/인코딩)
- 훈련/검증/테스트용으로 데이터 분할 등

훈련데이터로 모델 학습

다양한 알고리즘 적용

하이퍼 파라미터 튜닝을 통한 모델 성능 최적화

테스트 데이터로 모델의 성능 평가

- 모델 유형에 따른 다양한 평가지표 확인

분석모형 선정

▪ 머신러닝을 통해 해결하고자 하는 문제 정의 → 머신러닝 기법 설정

지도학습
Supervised Learning

레이블이 있는
훈련 데이터를 사용하여
모델을 학습시키는 방법

회귀 — 레이블이 연속형 수치인 경우 (예:집값,수요량 등)

분류 — 레이블이 범주형인 경우 (성별, 품종 등)

비지도학습
Unsupervised Learning

레이블이 없는
데이터에서
숨겨진 구조를 찾아내는 방법

군집화

차원축소

데이터 준비

- 훈련에 사용할 독립변수, 종속변수 선택

[아파트 가격 예측]

독립변수 (X, feature)			종속변수 (y, target, label)
면적	세대수	한강조망권 여부	아파트 가격
방의 갯수	건축연도	지하철역과의 거리	
화장실 갯수	배정 학교	공원과의 거리	

가격에 영향을 미칠 수 있는 요소

예측하려는 값

데이터 준비

▪ 훈련에 사용할 독립변수 선택

필터 방법(Filter Method)

- 특성의 통계적 특성에 기반하여 중요도를 평가하고, 특정 기준에 따라 변수를 선택하거나 제거
- 변수 선택을 모델 학습과 독립적으로 수행

상관계수 기반 선택 Correlation Coefficient	연속형 변수와 연속형 타겟 간의 상관 관계를 측정하여 중요한 특성을 선택
카이제곱 검정 기반 선택 Chi-square Test	범주형 변수와 타겟 같의 관계를 평가하여 선택
분산분석 기반 선택 Variance	분산이 낮은 특성(정보가 적은 특성)을 제거
정보이득 기반 선택 Information Gain	각 특성과 타겟 간의 정보의 양을 측정하여 선택

데이터 준비

▪ 훈련에 사용할 독립변수 선택

래퍼 방법(Wrapper Method)

- 특정 모델을 사용하여 다양한 특성 조합을 평가하고, 가장 좋은 성능을 보이는 조합을 선택
- 변수 선택 과정에서 모델 학습

전진선택법 Forward Selection	가장 중요한 특성부터 하나씩 추가하면서 모델의 성능을 평가
후진제거법 Backward Elimination	모든 특성으로 시작하여 가장 덜 중요한 특성부터 하나씩 제거
단계적 선택법 Stepwise Selection	전진선택법과 후진제거법을 번갈아 수행

데이터 준비

▪ 훈련에 사용할 독립변수 선택

임베디드 방법 (Embedded Method)

- 모델 학습 과정에서 변수 선택을 수행합니다. 특정 알고리즘이 특성 선택을 내장하고 있어 학습과 선택이 동시에 이루어진다.

L1 정규화 Lasso	회귀 모델에서 L1 정규화를 적용하여 특성 중 일부의 계수를 0으로 만들어 선택
L2 정규화 Ridge	L2 정규화를 통해 덜 중요한 특성의 영향력 감소
엘라스틱넷	Lasso와 Ridge의 조합
트리 기반 방법	트리 기반 모델은 특성 중요도를 계산할 수 있다. 이 중요도를 바탕으로 중요하지 않은 특성을 제거

데이터 준비

▪ 훈련에 사용할 독립변수 선택

차원 축소 방법 (Dimensionality Reduction)

- 차원 축소는 특성의 수를 줄이는 또 다른 접근 방식으로, 원래의 특성을 사용하지 않고 새로운 특성(주성분)을 생성한다.

주성분 분석 PCA	데이터의 분산을 최대화하는 새로운 축을 찾아 차원을 축소
선형 판별 분석 LDA	클래스 구분을 최대화하는 축을 찾는 방법으로, 특히 분류 문제에 유용

데이터 준비

- 훈련에 사용할 독립변수 선택

도메인 지식을 바탕으로

필터, 래퍼, 임베디드 방법 등 다양한 접근 방식을 조합하여

데이터와 문제의 특성에 맞는 최적의 변수를 선택

데이터 탐색 및 전처리

▪ 결측치 처리

결측치 제거	Listwise Deletion	결측치가 있는 행 전체 제거
	Pairwise Deletion	분석에 사용되는 변수에 결측치가 있는 경우만 제거
결측치 대체	특정 값으로 대체	(ex) 0으로 대체
	대푯값으로 대체	평균값, 중앙값, 최빈값 등으로 대체
	회귀 대체	결측치가 있는 변수를 다른 변수의 값으로 예측하여 대체
	KNN 대체	K-최근접 이웃 알고리즘을 사용하여 유사한 샘들들의 값으로 대체
	전방 채우기	이전 값으로 대체(시계열)
	후방 채우기	다음 값으로 대체(시계열)
	도메인 지식 활용	전문가의 판단에 따라 적절한 값으로 대체

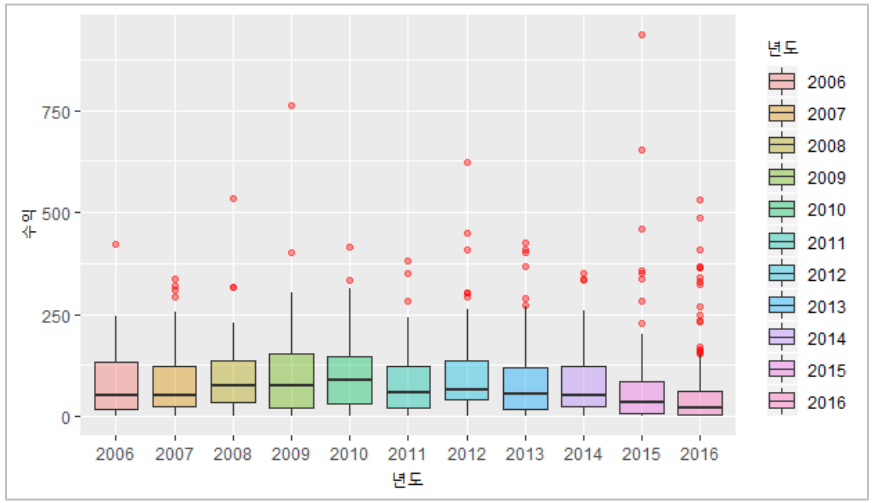
- 결측치 처리 방법에 따라 분석 결과가 달라질 수 있으므로 가능하면 여러 방법을 시도하고 결과를 비교하여 선택한다.

데이터 탐색 및 전처리

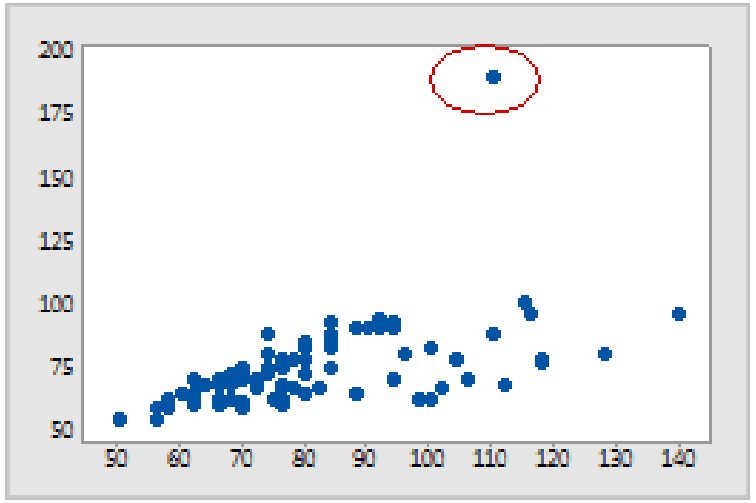
▪ 이상치 탐지

이상치 : 데이터에서 다른 데이터 포인트와 비교했을 때 비정상적으로 크거나 작은 값을 가진 관측치를 의미

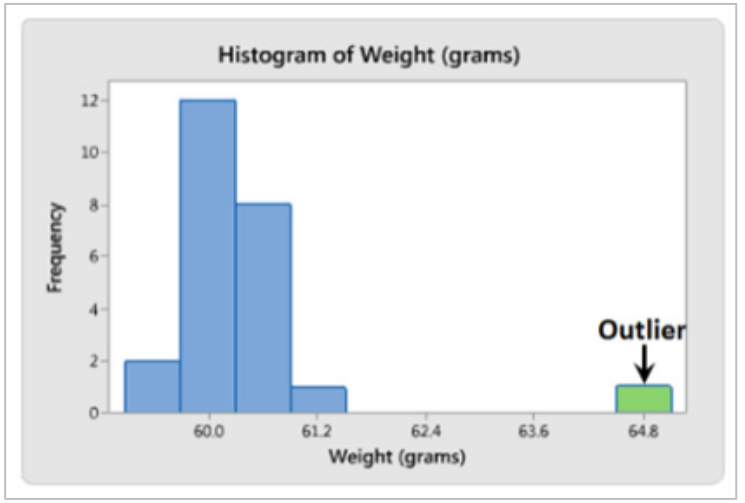
시각적 탐지



https://wikidocs.net/33920



https://support.minitab.com/ko-kr/minitab/help-and-how-to/graphs/general-graph-options/focus-on-critical-data/subsets-of-data/



https://blog.naver.com/jiehyunkim/220935734044

통계기반 탐지

- 사분위범위 기반(IQR)
- Z-SCORE : 각 데이터 포인트가 평균에서 얼마나 떨어져있는지를 표준편차 단위로 측정

데이터 탐색 및 전처리

▪ 이상치 처리

제거	• 확실한 이상치나 오류인 경우 해당 데이터 제거
변환	• 로그변환, 제곱근변환, Box-Cox 변환 등을 통해 큰 값의 영향을 줄임
대체	• 평균/중앙값으로 대체 • 이상치를 특정 한계값으로 대체 • 도메인 지식을 활용하여 수정
모델링 기법	• 이상치에 덜 민감한 모델링 기법 사용(랜덤포레스트, 부스팅모델 등)
스케일링	• 로버스트 스케일링 : 이상치의 영향을 줄이기 위해 중앙값과 IQR을 사용하여 데이터를 스케일링하는 방법

데이터 탐색 및 전처리

- 데이터 변환

변수의 데이터 유형 파악

데이터 유형		특징	예시
수치형 데이터	연속형 데이터	셀 수 없는 수치형 자료	키, 몸무게, 기온
	이산형 데이터	셀 수 있는 수치형 자료	나이, 판매량
범주형 데이터	순위형 데이터	순서가 있는 범주형 자료	등급, 학점
	명목형 데이터	순서가 없는 범주형 자료	성별, 혈액형, 지역

→ 범주형 데이터는 숫자형태로 인코딩 필요

데이터 탐색 및 전처리

- 데이터 변환

레이블 인코딩

문자를 숫자 형태로 변경

학점
A
B
C
D
F

→

학점
5
4
3
2
1

원핫 인코딩

더미변수 생성

계절
봄
여름
가을
겨울

→

봄	여름	가을	겨울
1	0	0	0
0	**1**	0	0
0	0	**1**	0
0	0	0	**1**

데이터 탐색 및 전처리

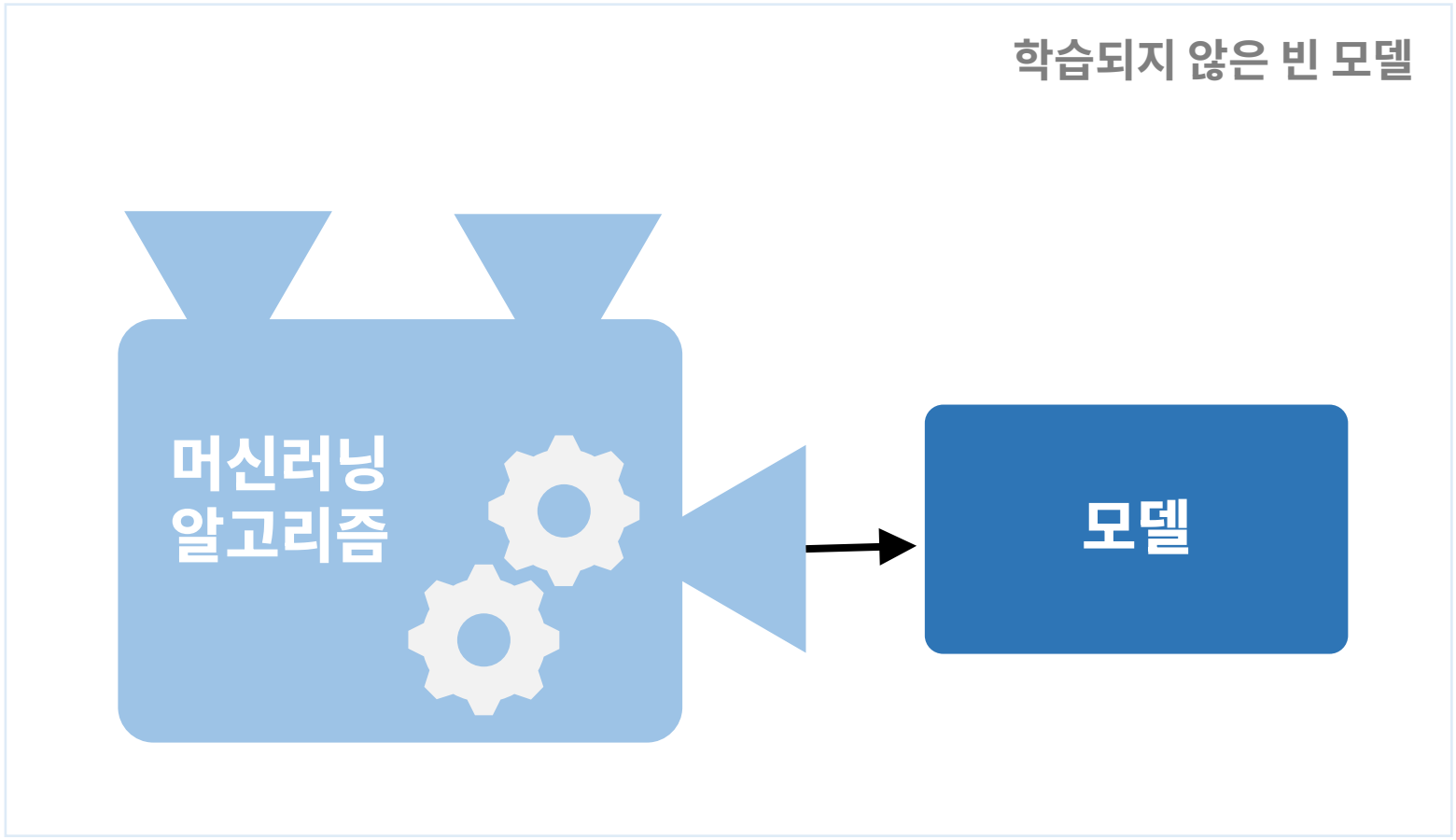
▪ 피처 스케일링(feature scaling)

- 특성(변수)의 값이 서로 다른 범위에 있는 경우, 이를 동일한 범위로 조정하는 방법
- 수치형 변수의 범위/분포가 차이나는 경우 사용

종류	방법	적용 사례
표준화 (Standardization)	**평균이 0, 분산이 1**이 되도록 변환	• 스케일에 민감한 알고리즘(선형회귀, 로지스틱회귀, SVM 등) • 데이터가 정규분포를 따른다고 가정하는 경우
정규화 (Normalization)	데이터를 **0과 1 사이의 범위**로 변환 (음수값이 있는 경우 -1~1 사이로 변환)	• 거리기반 알고리즘을 사용하는 경우(KNN, SVM 등) • 데이터가 정규분포를 따르지 않는 경우
로버스트 스케일링 (Robust Scaling)	**중위수(median)와 IQR(사분위수 범위)**를 사용하여 스케일링	• 이상치가 존재할 때 평균과 표준 편차 대신 중위수와 IQR을 사용하여 **이상치의 영향을 줄인다.**
로그 변환 (Log Transformation)	데이터의 값을 **로그 스케일**로 변환	• **비대칭 분포**: 데이터가 정규분포에서 벗어나 한쪽으로 치우쳐 있는 경우 로그 변환을 통해 분포를 정규화할 수 있다. • 로그 변환은 **양수 값에 대해서만 가능**하다.

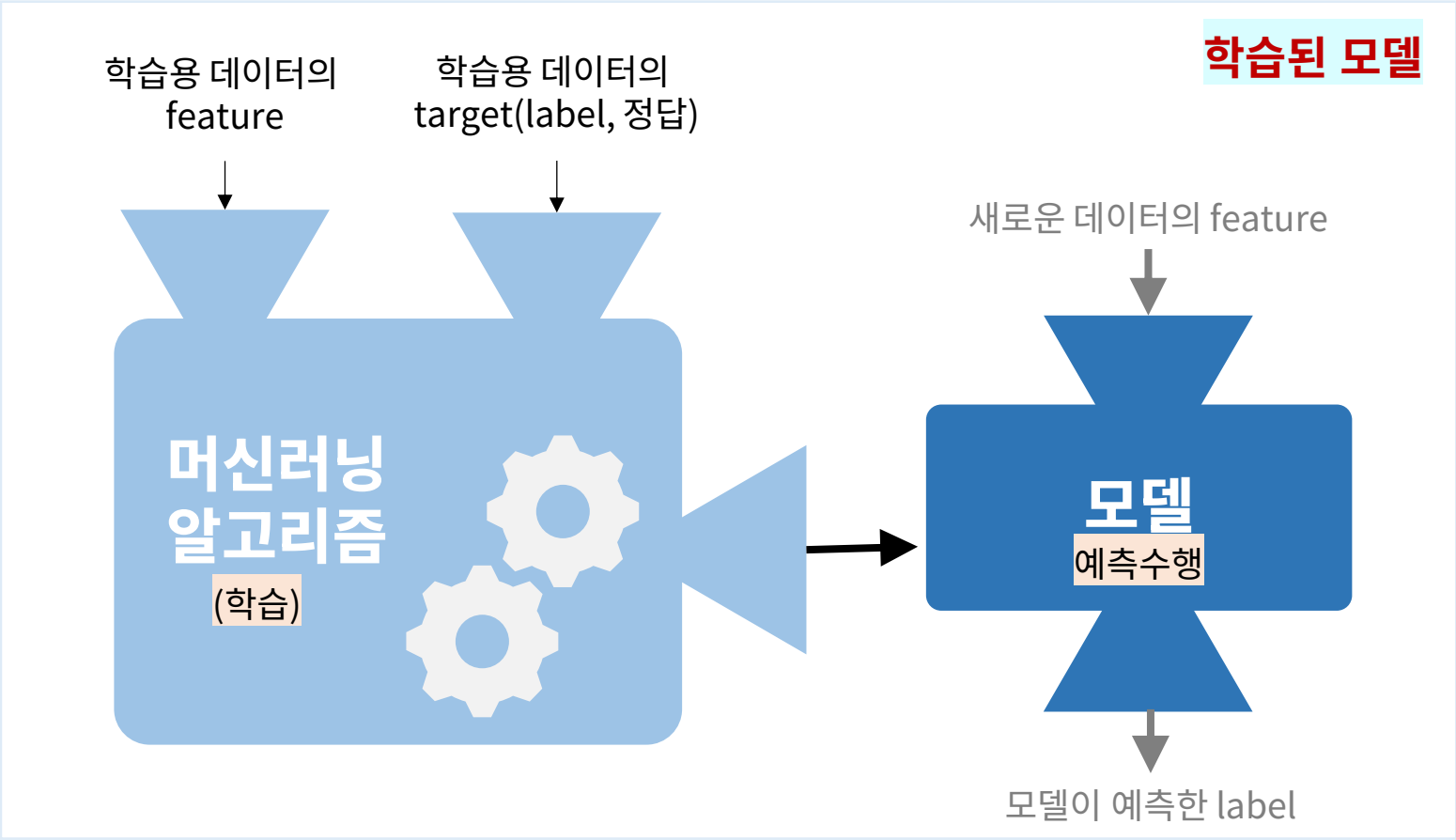
모델 학습 및 평가

- 모델 객체 생성



모델 학습 및 평가

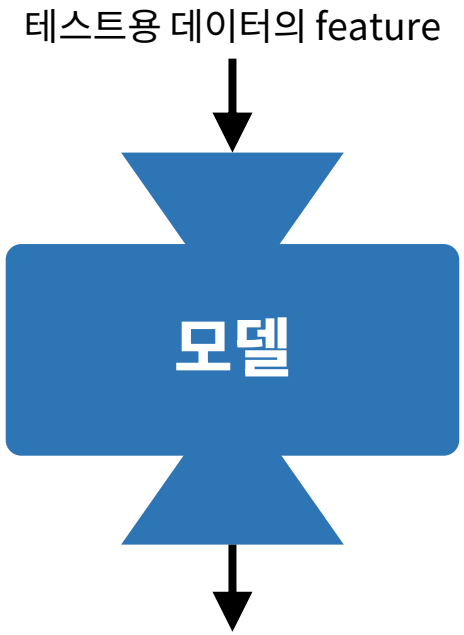
▪ 모델 학습



모델 학습 및 평가

- 예측

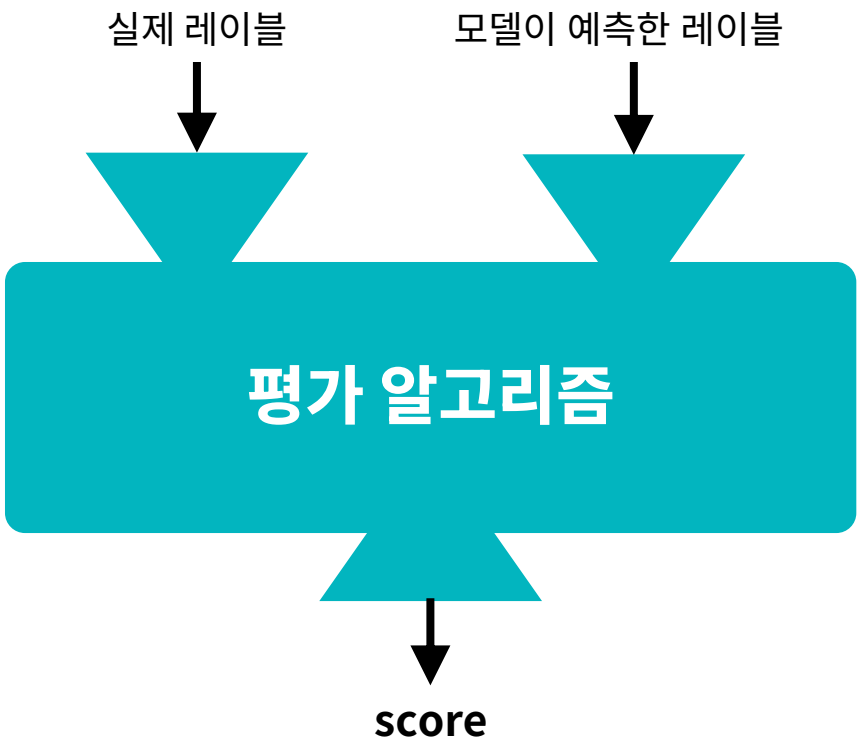
  - 테스트 데이터의 features를 모델에 입력하여 레이블 예측



모델 학습 및 평가

- 모델의 정확도 평가

  - 테스트 데이터의 레이블과 모델이 예측한 레이블을 이용하여 평가



모델 학습 및 평가

▪ 모델의 평가 지표

회귀

- MSE
- MAE
- RMSE
- R^2

분류

- Accuracy
- Precision
- Recall
  Sensitivity
- F1 Score
- ROC 곡선

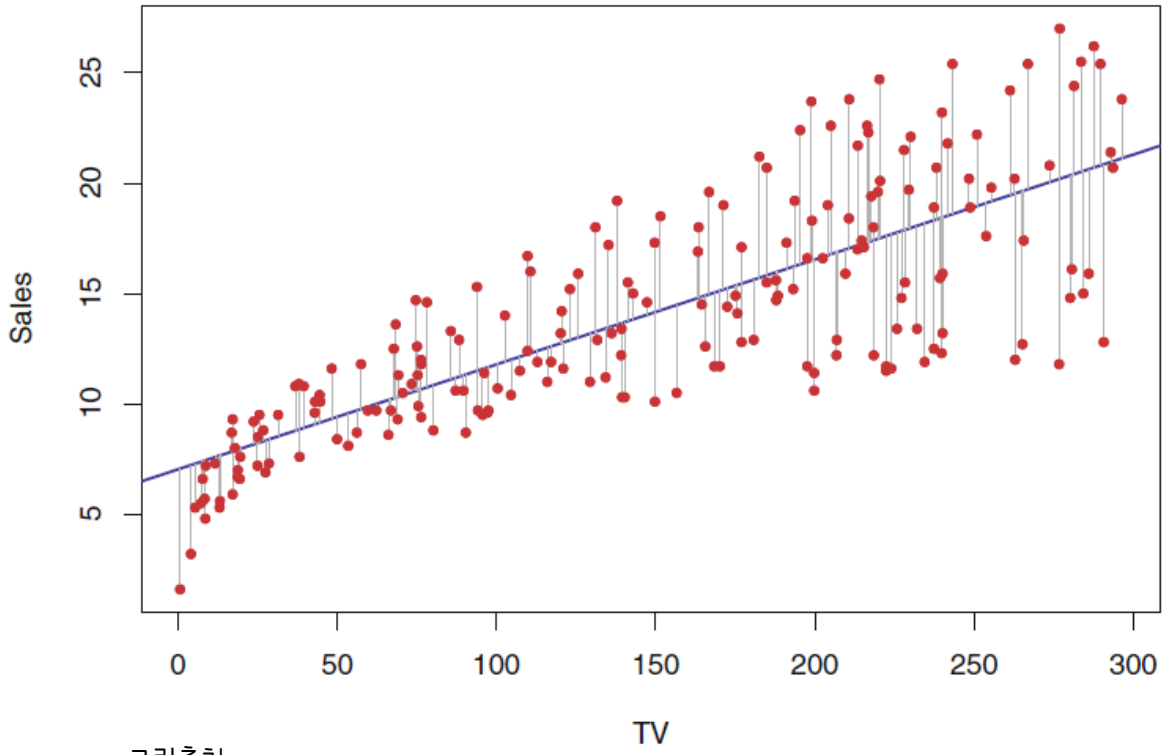
군집화

- 실루엣 계수
  (Silhouette
  Coefficient)

LinearRegression

단순선형회귀

- 하나의 독립변수(X)와 하나의 종속변수(y) 사이의 선형 관계를 모델링



그림출처
https://godongyoung.github.io/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D/2018/01/20/ISL-linear-regression_ch3.html

목표 : 데이터에 가장 잘 맞는 예측선을 찾는것.

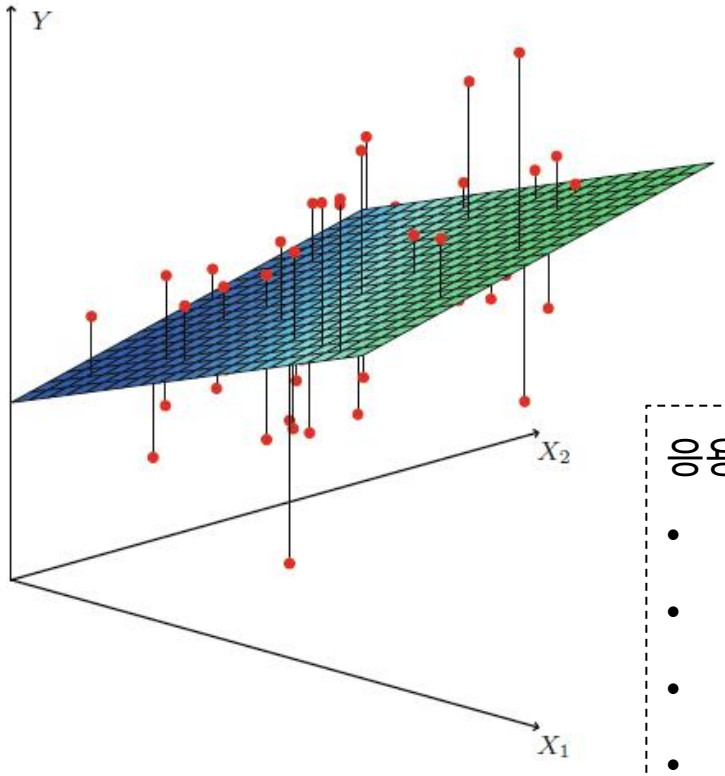
$$h(x) = wx + b$$

응용예시

- **경제학**: 특정 제품의 가격과 판매량 간의 관계 분석
- **의학**: 환자의 나이와 특정 질병 발병률 간의 관계 분석
- **사회과학**: 교육 수준과 소득 수준 간의 관계 분석

다중회귀

▪ 2개 이상의 독립변수와 하나의 종속변수 사이의 관계를 모델링



목표 : 여러 독립 변수와 종속 변수 사이의 관계를 가장 잘 설명하는 평면 또는 초평면을 찾는 것.

$$h(x) = w_1 x_1 + w_2 x_2 + \cdots + b$$

응용예시

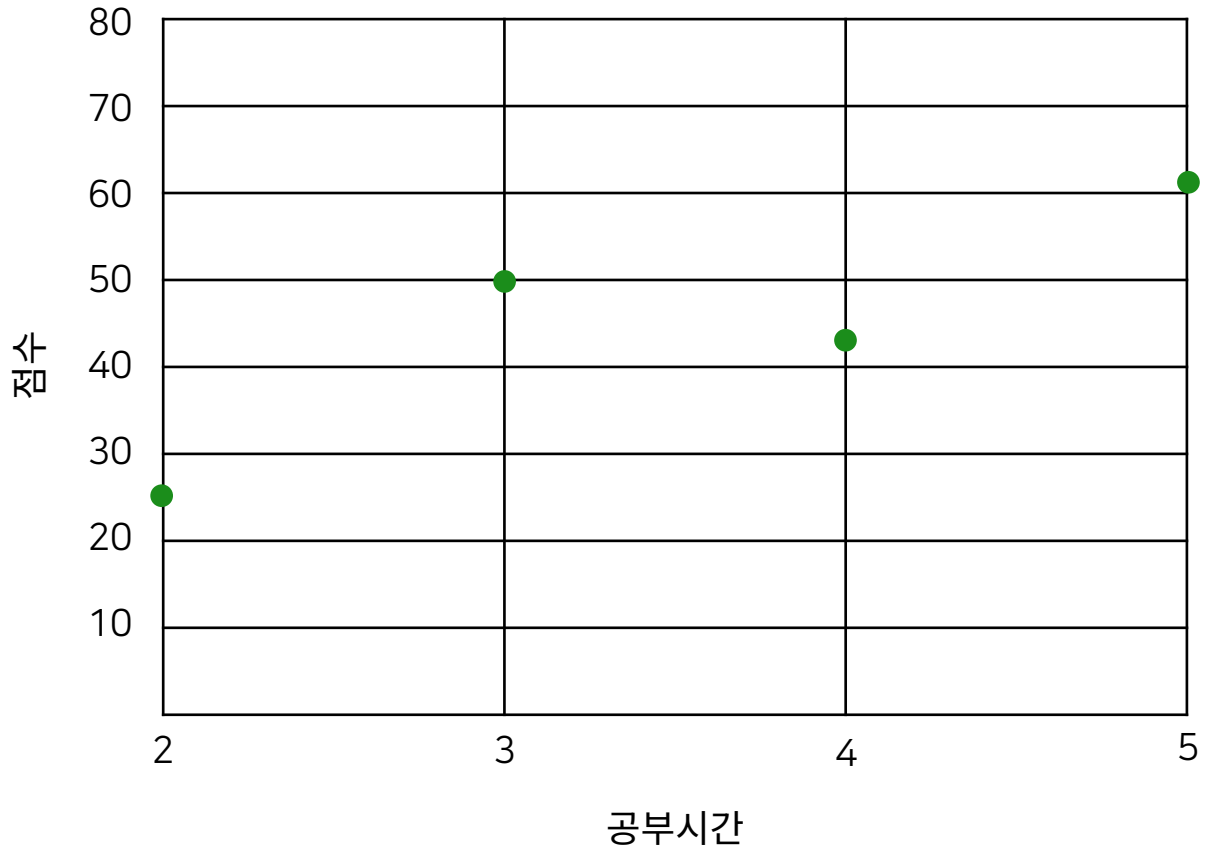
- 부동산 : 집의 크기, 위치, 연식 등을 이용하여 집의 가격 예측
- 경제학 : 소비자 지출, 소득, 세금 등의 여러 요인을 고려하여 경제성장률 예측
- 의학 : 다양한 환자 정보(나이, 성별, 생활습관 등)를 사용하여 질병의 발병 확률 예측
- 보건 : 부모 키에 따른 자녀의 키 예측

그림출처
https://godongyoung.github.io/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D/2018/01/20/ISL-linear-regression_ch3.html

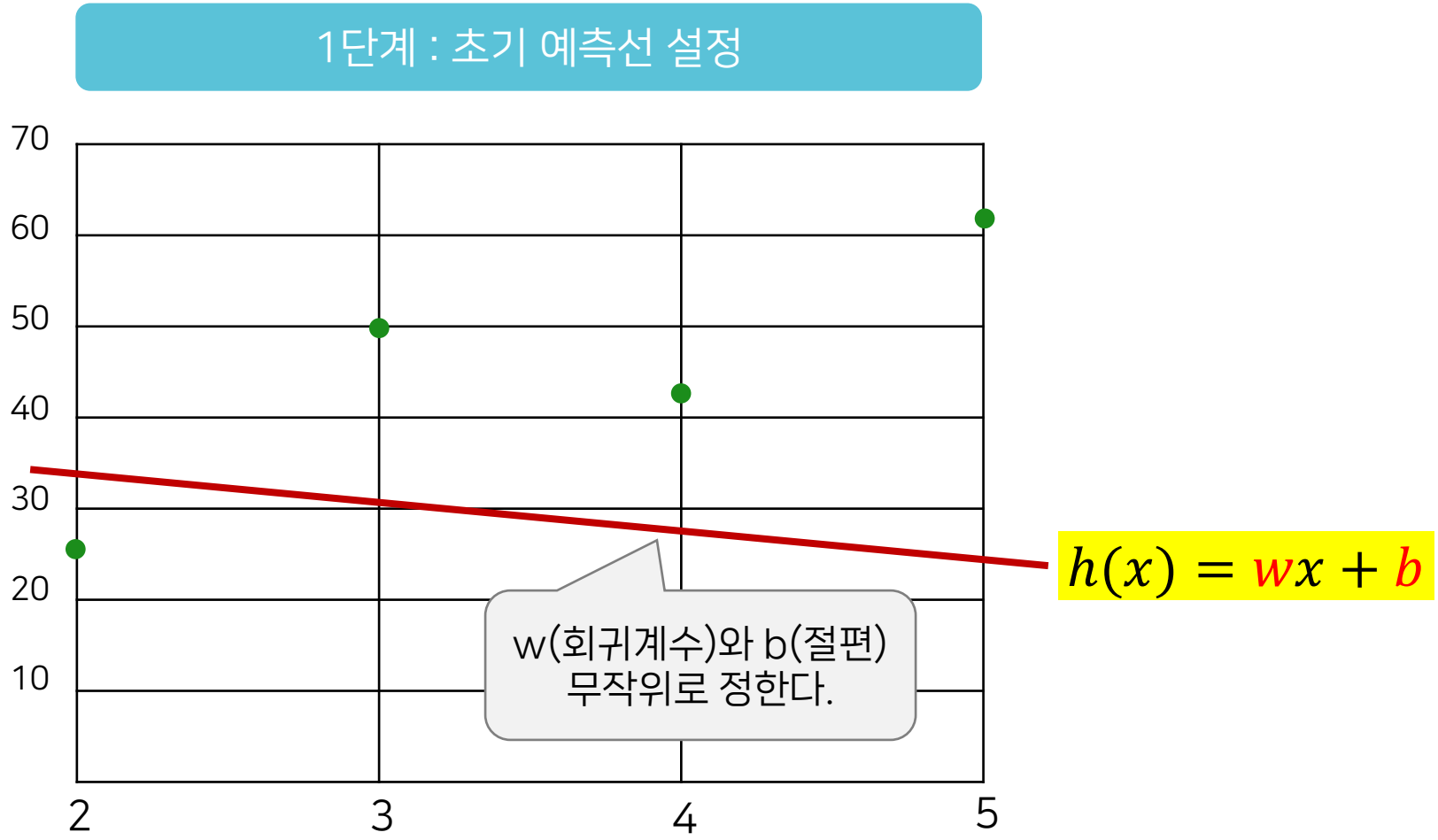
선형회귀 동작원리

- 공부한 시간에 따른 점수를 예측하기위한 함수를 찾는 방법

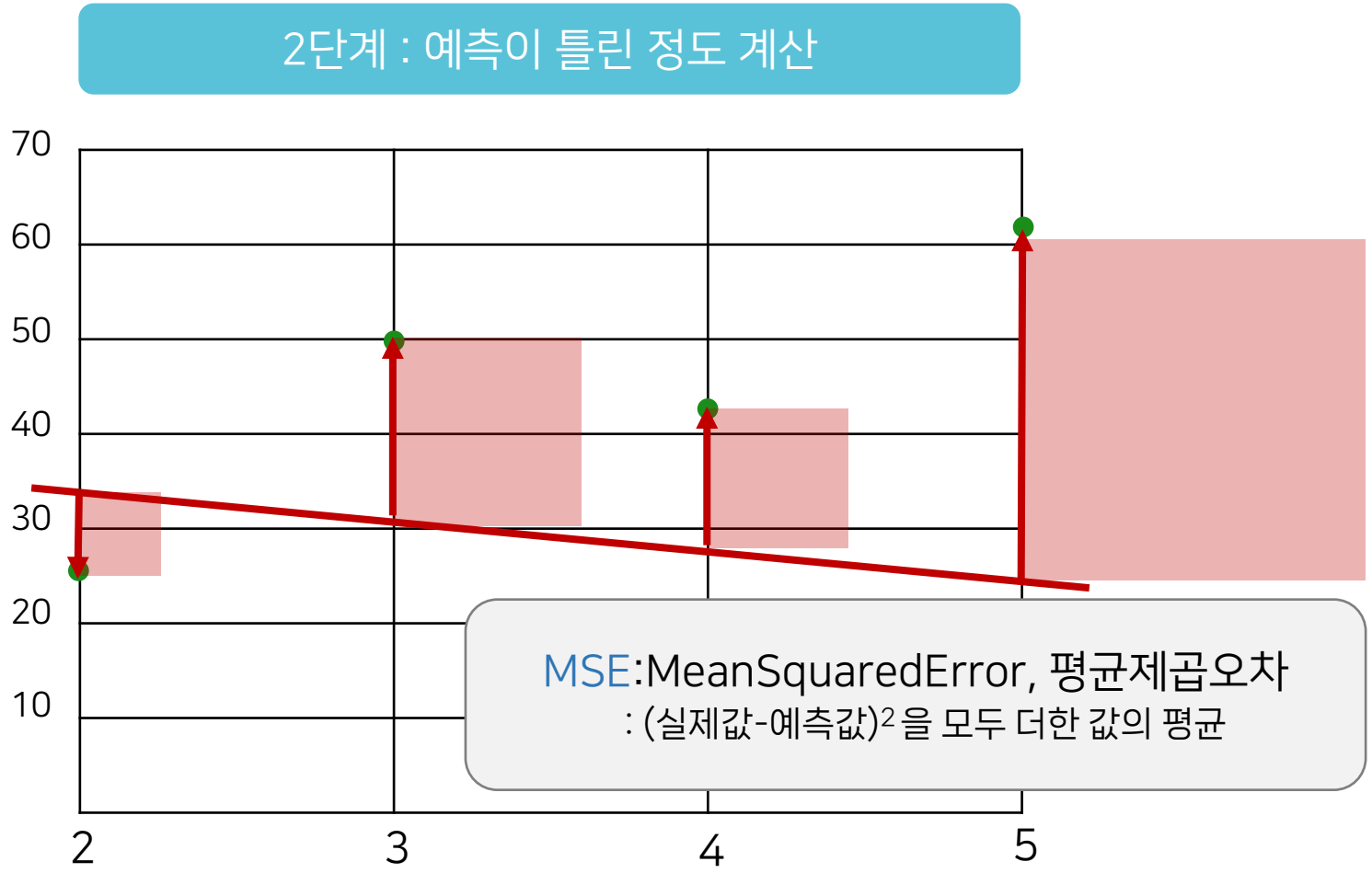
공부시간	점수
2	25
3	50
4	42
5	61



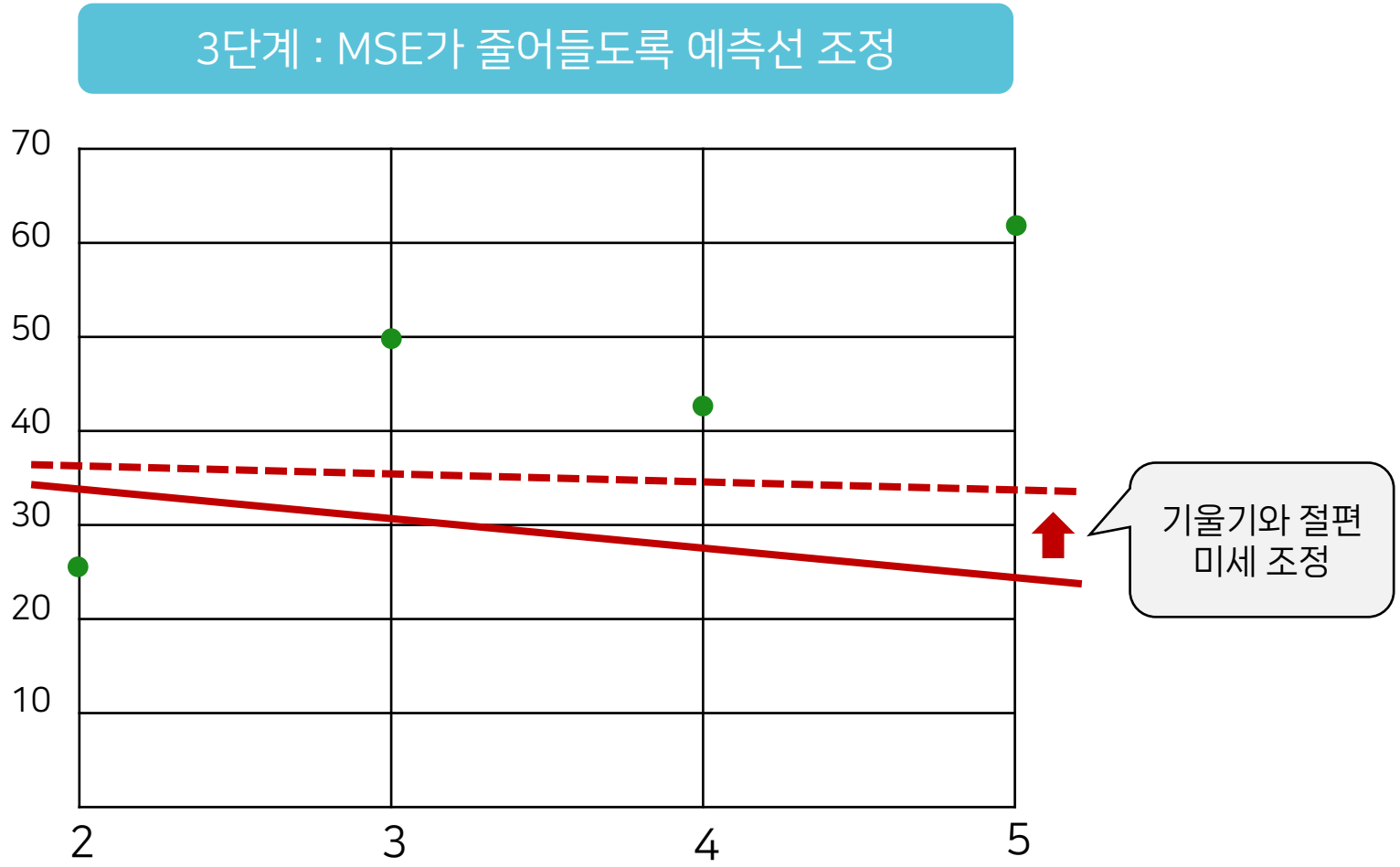
선형회귀 동작원리



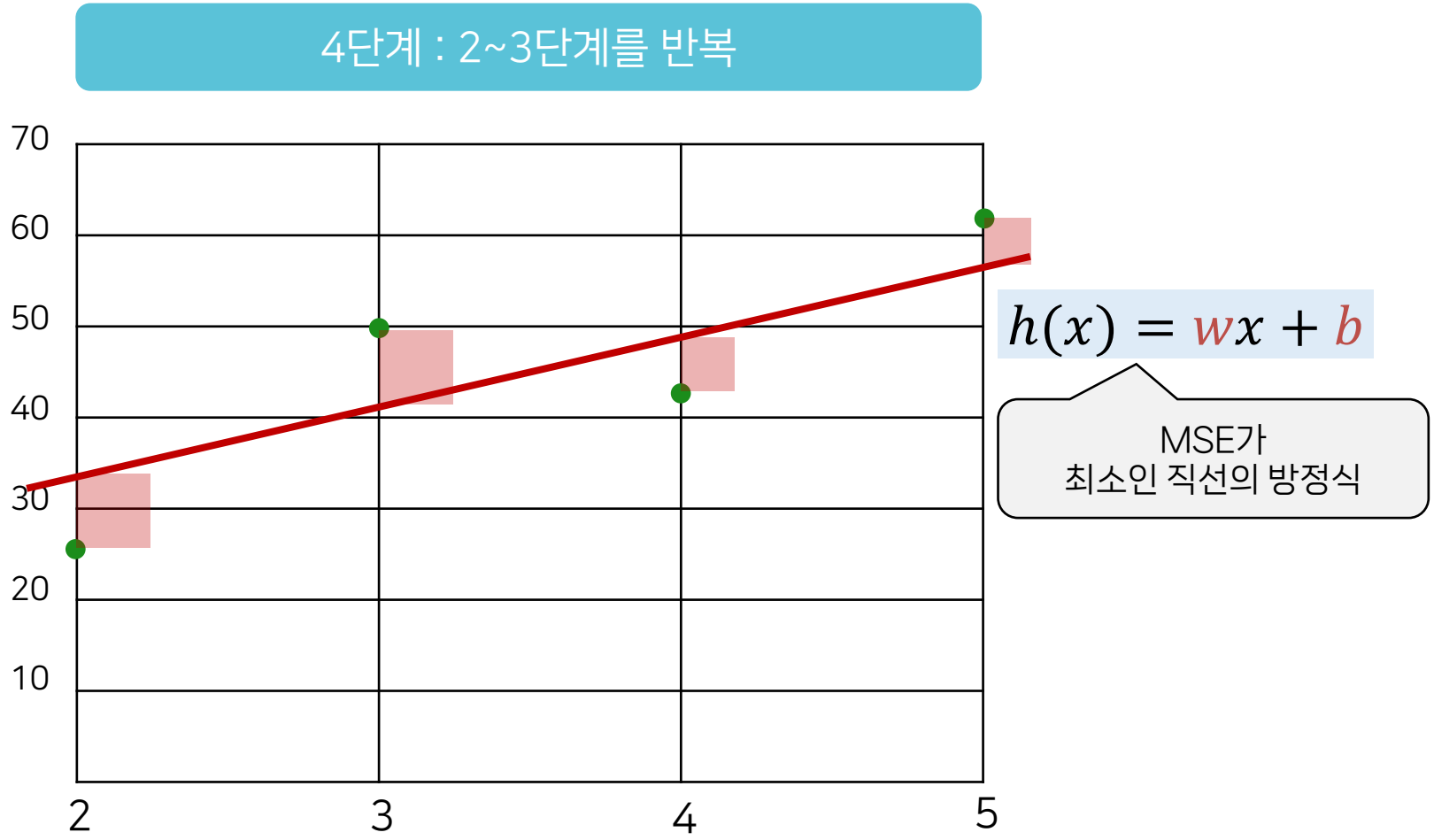
선형회귀 동작원리



선형회귀 동작원리



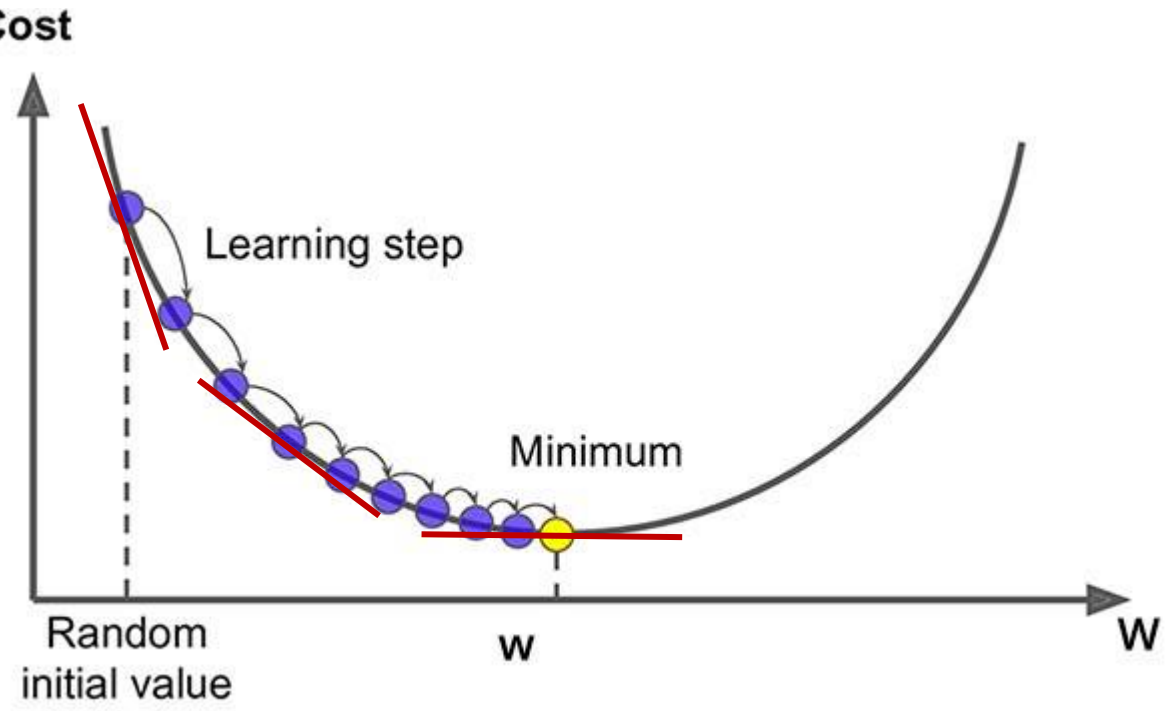
선형회귀 동작원리



선형회귀 동작원리

▪ 경사하강법(Gradient Descent)

- 반복적인 계산을 통해 w값을 업데이트 하면서 MSE가 최소가 되는 w값을 구하는 방법
- 미분값이 감소하는 방향으로 순차적으로 w를 업데이트한다.



https://mlpills.dev/machine-learning/gradient-descent/

- 미분했을 때 기울기가 +이면 음의 방향으로 이동
- 미분했을 때 기울기가 -이면 양의 방향으로 이동
- 미분값이 최소인 지점을 찾는다.

회귀 모델의 성능 평가 지표

Loss

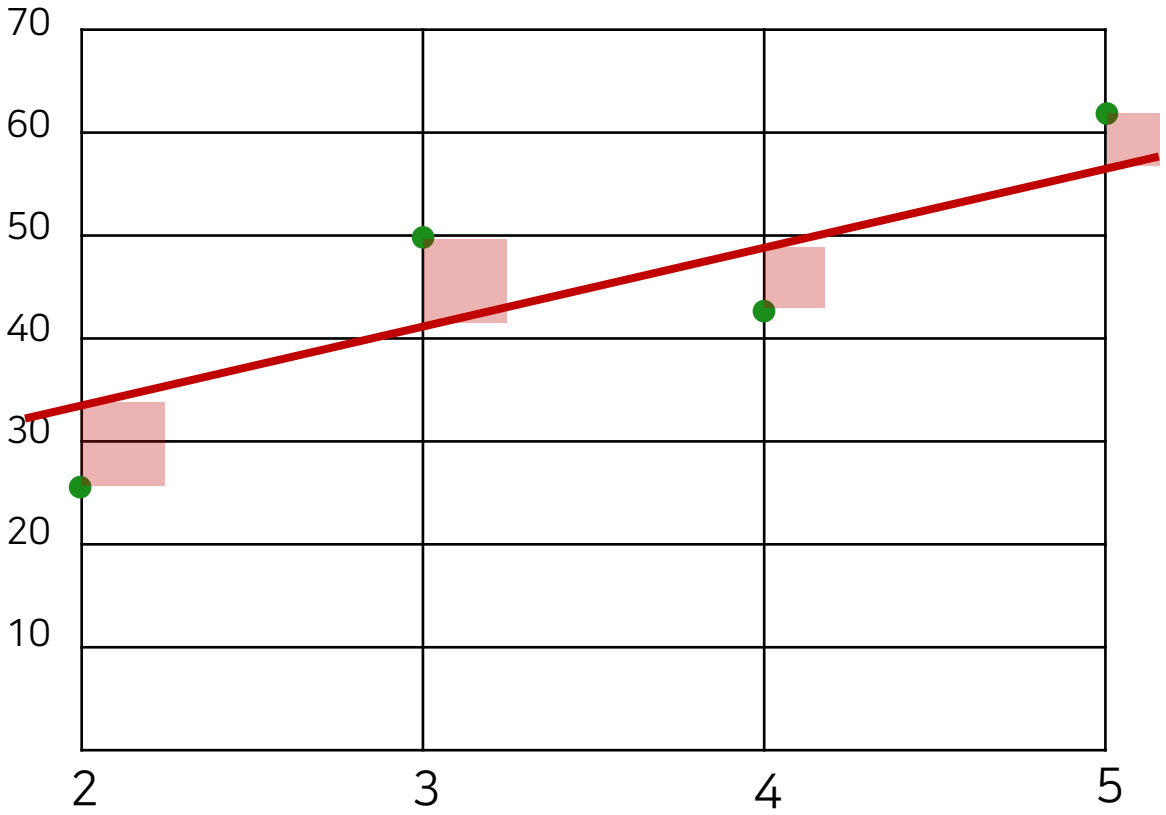
실제 값과 모델이 예측하는 값의 차이에 기반한 평가 방법 사용

MSE	Mean Squared Error
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
R^2	R-squared. 결정계수(0~1 사이의 값)

회귀 모델의 성능 평가 지표

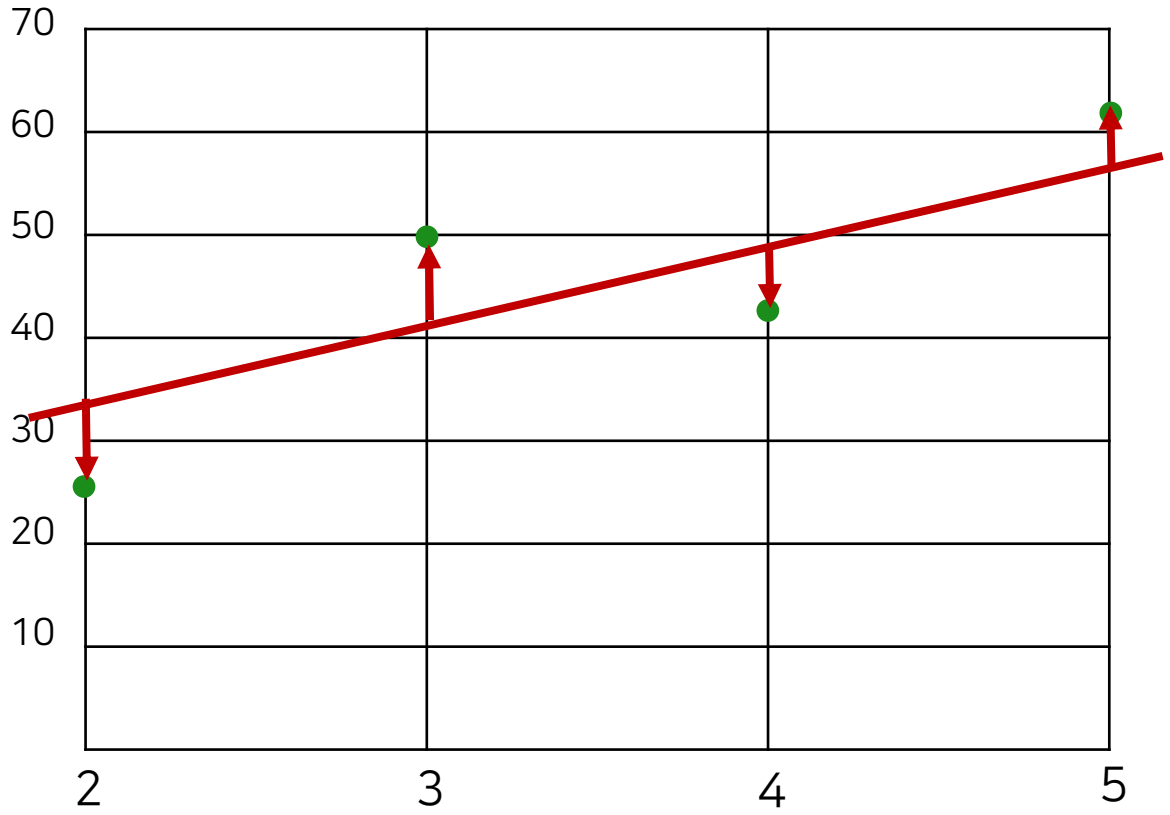
MSE Mean Squared Error, 평균제곱오차

타깃과 예측의 차이를 제곱한 것들의 합의 평균



MAE Mean Absolute Error, 평균절대값오차

실제값과 예측값의차이의 절대값의 합의 평균

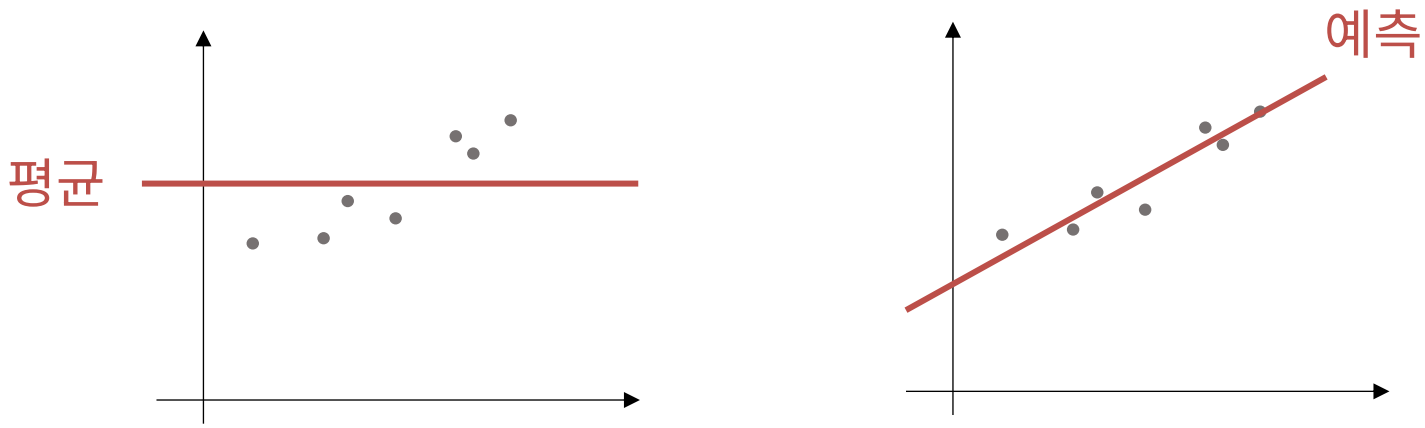


회귀 모델의 성능 평가 지표

R^2 R-squared. 결정계수 → 1에 가까울수록 높은 성능의 모델이라고 할 수 있음

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{(\text{타깃} - \text{예측})^2\text{의 합}}{(\text{타깃} - \text{평균})^2\text{의 합}}$$

잘 근사할수록 0 에 가까움



TSS : 타깃과 평균의 차이를 제곱한 것들의 합 > RSS : 타깃과 예측의 차이를 제곱한 것들의 합

광고 플랫폼에 따른 판매량 예측

- 단순선형회귀
- 다중회귀
- 회귀모형의 평가지표
- 다항회귀
- 규제

문제 정의

광고 플랫폼 별 광고비에 따른 판매량 분석

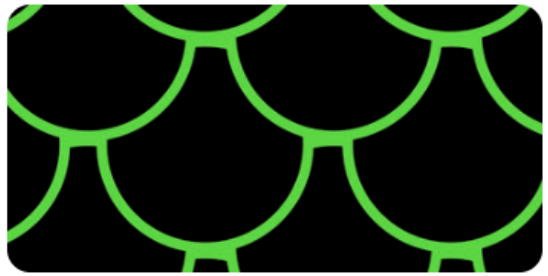
- TV 광고비에 따른 판매량은 어떻게 될까?
- Radio 광고비에 따른 판매량은 어떻게 될까?
- Newspaper 광고비에 따른 판매량은 어떻게 될까?

요즘으로 치면?
✓ SNS 유형에 따른 판매량 예측 (인스타, 유튜브, 페이스북, 블로그 등)
✓ 인스타그램 팔로워 수에 따른 판매량 예측

데이터 수집

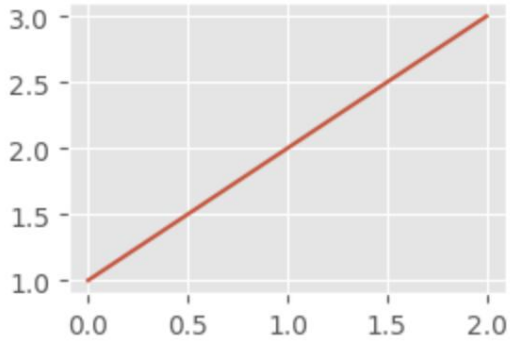
https://www.kaggle.com/datasets/ashydv/advertising-dataset

Advertising Dataset



라이브러리 불러오기

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

plt.style.use('ggplot')
pd.Series([1,2,3]).plot(figsize=(3,2))
```



데이터 준비

- 데이터 불러오기

```
df = pd.read_csv('data/advertising.csv')
df.head()
```

```
df.head()
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

✓ TV, Radio, Newspapaper의 기본 단위 : 1,000달러($)

✓ Sales의 기본 단위 : 1,000개

데이터 준비

- 데이터 정보 확인

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

샘플의 수

컬럼 수

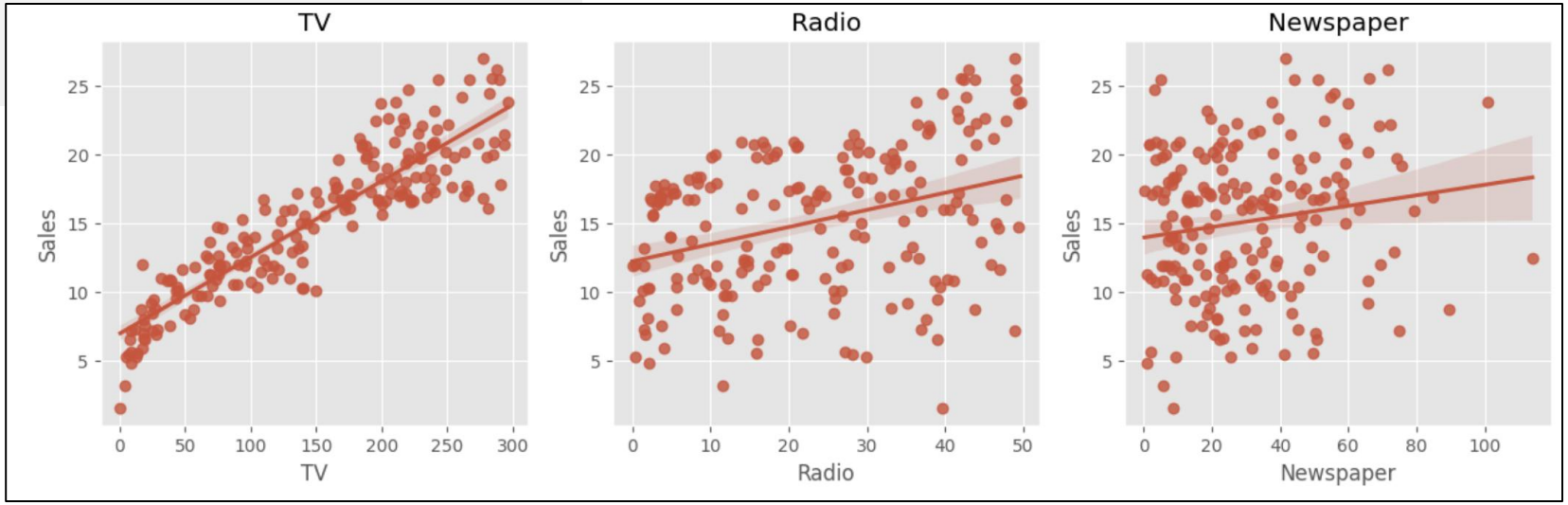
결측치 없음

모두 수치형

데이터 이해

- 매체별 광고비에 따른 판매량 시각화

```
# 그래프 크기
plt.figure(figsize=(15,4))

for i, feature_name in enumerate(df.columns[:3]):
    plt.subplot(1,3,i+1)
    sns.regplot(data=df, x=feature_name, y='Sales')
    plt.title(feature_name)

# 그래프 보이기
plt.show()
```



데이터 이해

▪ 상관계수

```
sns.heatmap(df.corr(), cmap='Blues', annot=True)
plt.show()
```

예측하고자 하는 종속변수와
선형관계가 강한 독립변수를 찾는다.

Sales와 가장 선형관계가 강한 매체는?

TV

데이터 이해

▪ 피어슨 상관계수

- 피어슨 상관계수란?
  - ✓ 두 변수 간의 선형 관계 강도와 방향을 나타내는 통계적 지표
  - ✓ -1~1 사이의 값을 가진다.
  - ✓ 산점도를 통해 시각적으로 확인 가능하다.
  - ✓ 1에 가까울수록 점들이 직선에 가깝게 분포

데이터 이해

▪ 피어슨 상관계수의 해석

- **상관관계의 강도**
  - ✓ **절댓값**이 1에 가까울수록 강한 상관관계
  - ✓ 절댓값이 0에 가까울수록 약한 상관관계
- 상관관계의 방향
  - ✓ 양수 : 양의 상관관계(독립변수가 증가할 때 종속변수도 증가)
  - ✓ 음수 : 음의 상관관계(독립변수가 증가할 때 종속변수는 감소)
- 일반적인 해석 기준(절댓값 기준)
  - ✓ 0 : 상관관계 없음
  - ✓ ~0.19 : 매우 약한 상관관계
  - ✓ ~0.39 : 약한 상관관계
  - ✓ ~0.59 : 중간 정도의 상관관계
  - ✓ ~0.79 : 강한 상관관계
  - ✓ ~0.99 : 매우 강한 상관관계
  - ✓ 1 : 완벽한 선형관계

Linear Regression - 단순선형회귀

- 변수 선택

```
X = df[['TV']] # 독립변수는 2차원이어야 함
y = df['Sales']
```

독립변수 X

종속변수 y

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

Linear Regression - 단순선형회귀

- 훈련세트, 테스트세트 분할

```
# 훈련세트, 테스트세트 분할
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(150, 1) (50, 1) (150,) (50,)
```

- **sklearn.model_selection** : 데이터 분할 및 검증과 관련된 도구들을 제공하는 모듈
- **train_test_split** : 학습용 데이터와 테스트용 데이터로 분할하기 위한 함수
- **test_size** : 테스트용 데이터 비율. 지정하지 않으면 디폴트인 0.25가 적용된다.(option)
- **random_state** : 매번 동일한 결과를 얻기 위해 지정하였다.(option)

Linear Regression - 단순선형회귀

- 훈련세트, 테스트세트 분할 확인

```
# 훈련세트, 테스트세트 분할 확인
sns.scatterplot(x=X_train['TV'], y=y_train, label="train")
sns.scatterplot(x=X_test['TV'], y=y_test, label="test")
plt.legend()
plt.title("train_test split")
plt.xlabel('TV')
plt.show()
```

Linear Regression - 단순선형회귀

- 모델 생성 및 훈련

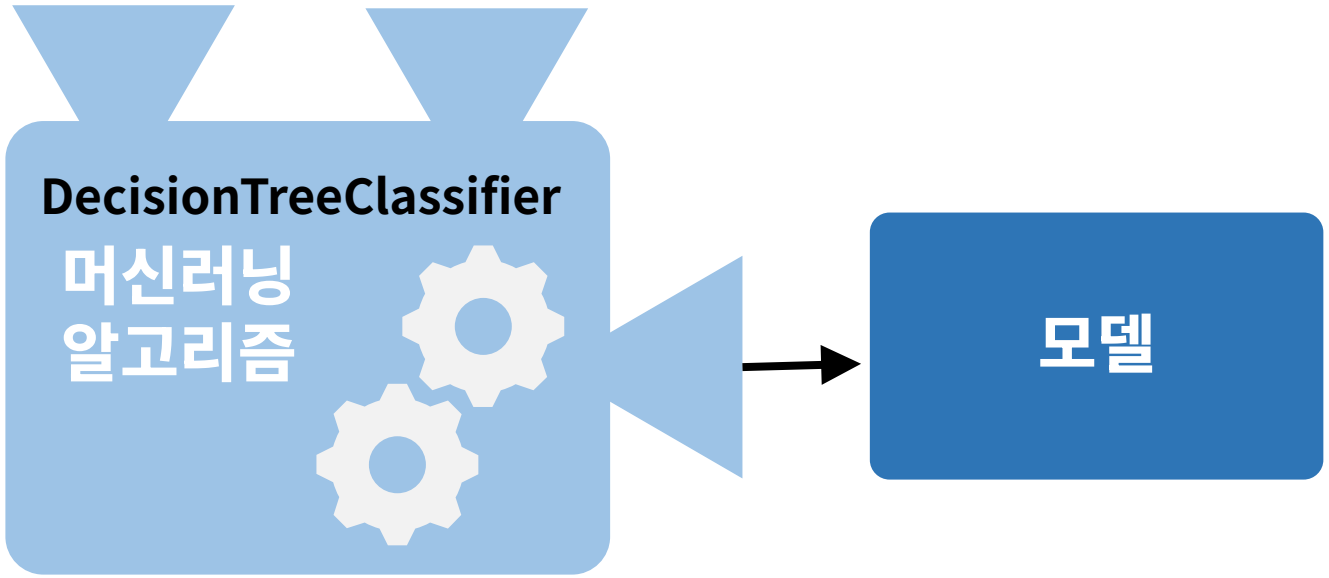
```
# LinearRegression 클래스 import
from sklearn.linear_model import LinearRegression

# 모델 생성하기
model = LinearRegression()

# 훈련데이터로 모델 훈련하기
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Linear Regression - 단순선형회귀

- 모델 생성

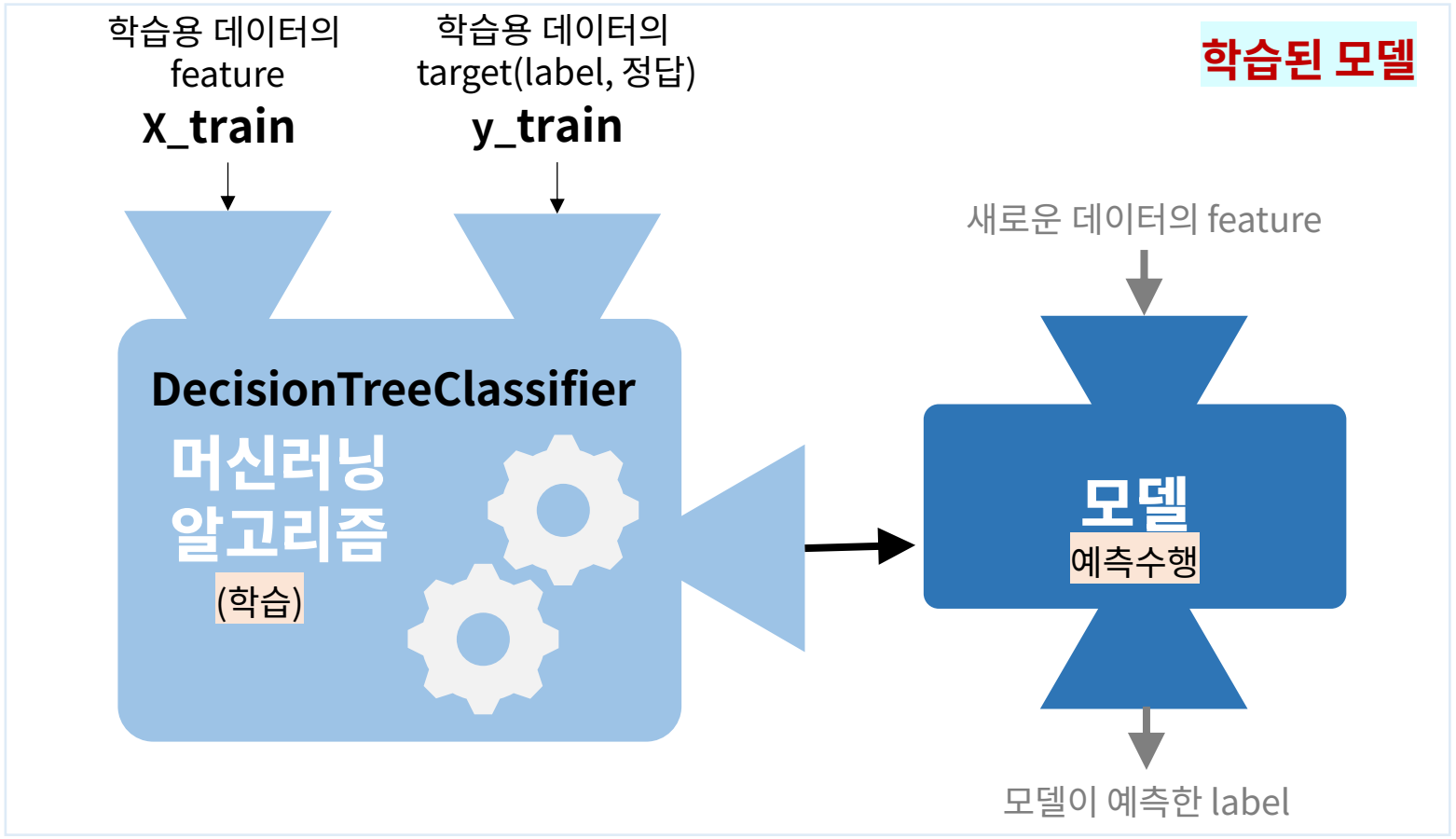
학습되지 않은 빈 모델




```
model = LinearRegression()
```

Linear Regression - 단순선형회귀

- 모델 학습




```
model.fit(X_train, y_train)
```

Linear Regression - 단순선형회귀

- 테스트세트로 평가

```
# 평가 방법 선택 : MSE, R2
from sklearn.metrics import mean_squared_error, root_mean_squared_error, mean_absolute_error, r2_score

# 테스트 데이터의 예측값
pred = model.predict(X_test)

# 예측값과 실제값의 차이(Loss/Error) 이용한 모델 평가
mse = mean_squared_error(y_test, pred)
rmse = root_mean_squared_error(y_test, pred)
mae = mean_absolute_error(y_test, pred)
r2 = r2_score(y_test, pred)

print(f'MSE : {mse:.2f}')
print(f'RMSE: {rmse:.2f}')
print(f'MAE : {mae:.2f}')
print(f'R2  : {r2:.2f}')
```

```
MSE : 5.51
RMSE: 2.35
MAE : 1.87
R2 : 0.80
```

Linear Regression - 단순선형회귀

- 예측

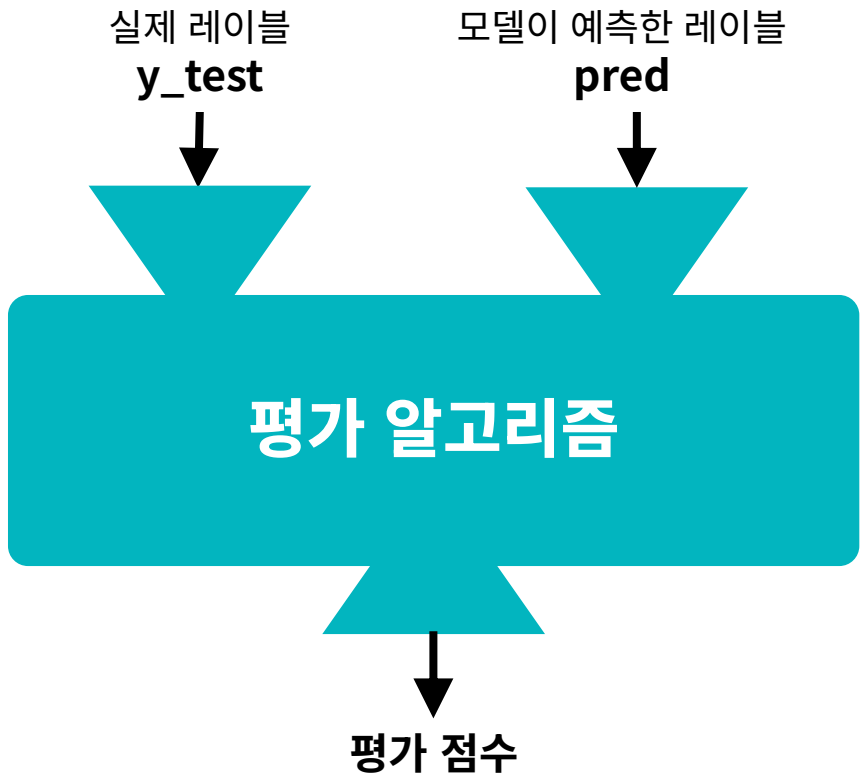
  - 테스트 데이터의 features를 모델에 입력하여 레이블 예측


```
pred = model.predict(X_test)
```

Linear Regression - 단순선형회귀

- 모델 성능 평가

  - 테스트 데이터의 레이블과 모델이 예측한 레이블을 이용하여 평가




```
mse = mean_squared_error(y_test, pred)
rmse = root_mean_squared_error(y_test, pred)
mae = mean_absolute_error(y_test, pred)
r2 = r2_score(y_test, pred)
```

Linear Regression - 단순선형회귀

- 실제값과 모델의 예측값 비교

```
df_pred = pd.DataFrame({'X_test':X_test['TV'],
                        'y_test':y_test,
                        'y_pred':pred})

df_pred['error'] = df_pred['y_test']-df_pred['y_pred']
df_pred.head(10)
```

	X_test	y_test	y_pred	error
95	163.3	16.9	16.143474	0.756526
15	195.4	22.4	17.921382	4.478618
30	292.9	21.4	23.321569	-1.921569
158	11.7	7.3	7.746876	-0.446876
128	220.3	24.7	19.300506	5.399494
115	75.1	12.6	11.258382	1.341618
69	216.8	22.3	19.106654	3.193346
170	50.0	8.4	9.868180	-1.468180
174	222.4	16.5	19.416818	-2.916818
45	175.1	16.1	16.797035	-0.697035

Linear Regression - 단순선형회귀

- 회귀 모델의 평가점수 수동 계산

```
mae = abs(df_pred['error']).mean()
mse = np.square(df_pred['error']).mean()
rmse = np.sqrt(mse)

rss = np.square(df_pred['y_test'] - df_pred['y_pred']).sum()
tss = np.square(df_pred['y_test'] - df_pred['y_test'].mean()).sum()
r2 = 1 - (rss/tss)

print(f'mse:{mse:.2f}')
print(f'rmse:{rmse:.2f}')
print(f'mae:{mae:.2f}')
print(f'r2:{r2:.2f}')
```

```
mse:5.51
rmse:2.35
mae:1.87
r2:0.80
```

Linear Regression - 단순선형회귀

- 모델의 학습 결과

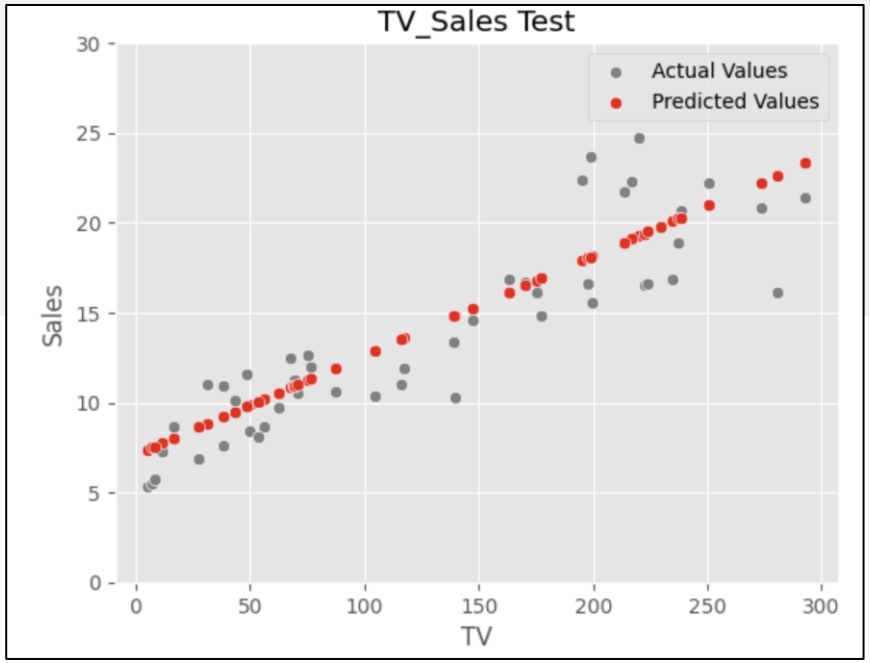
```
print(f'회귀계수:{model.coef_}')
print(f'절편:{model.intercept_}')
print(f'회귀식:{model.coef_[0]}  TV + {model.intercept_}')
```

```
회귀계수:[0.05538653]
절편:7.098853680118275
회귀식:0.05538653085519158  TV + 7.098853680118275
```

Linear Regression - 단순선형회귀

- 모델의 학습 결과

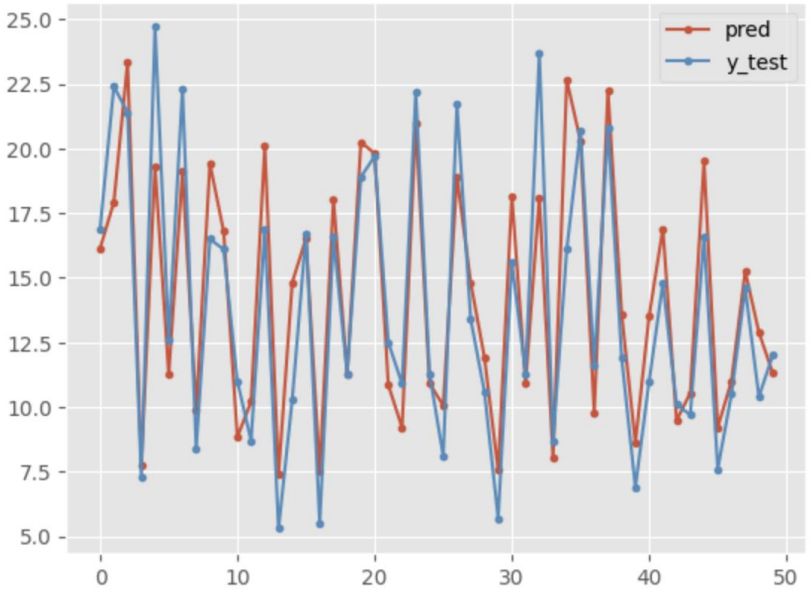
```
# 테스트데이터 실제값과 예측값 확인
sns.scatterplot(x=X_test['TV'], y=y_test, color='gray', label='Actual Values')# 테스트데이터 실제값
sns.scatterplot(x=X_test['TV'], y=pred, color='red', label='Predicted Values') # 테스트데이터 예측값
plt.xlabel('TV')
plt.ylim(0,30)
plt.title('TV_Sales Test')
plt.show()
```



Linear Regression - 단순선형회귀

- 모델의 학습 결과

```
# 테스트데이터 실제값과 예측값 확인
plt.plot(pred, marker='.', label='pred')
plt.plot(np.array(y_test), marker='.', label='y_test')
plt.legend()
```



Linear Regression - 단순선형회귀

- 새로운 값 예측하기

```
# TV 광고비 입력받기
tv = 100
print(f'TV광고비 : {tv}')

# 예측 함수 사용
print(model.predict([[tv]]))

# 모델이 학습한 공식에 대입(wx+b)
print(model.coef_ * tv + model.intercept_)
```

```
TV광고비 : 100
[12.63750677]
[12.63750677]
```

Linear Regression - 다중회귀

- 독립변수, 종속변수 선택

```
# 독립변수, 종속변수 선택
X2 = df[['TV','Radio','Newspaper']]
y = df['Sales']
```

독립변수 X

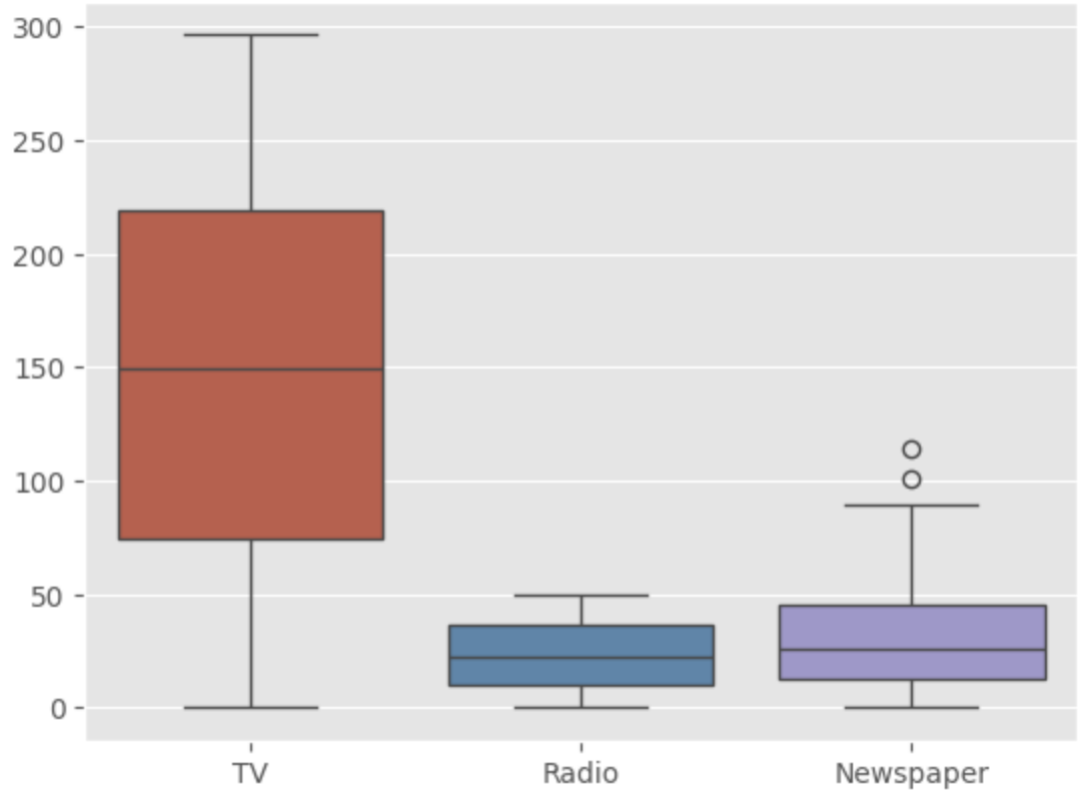
종속변수 y

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

Linear Regression - 다중회귀

- 독립변수, 종속변수 선택

```
# 독립변수의 데이터 분포 파악
sns.boxplot(X2)
```



Linear Regression - 다중회귀

▪ 데이터 스케일링

```
# 데이터 스케일링
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# scaler.fit(X2)
# X_scaled = scaler.transform(X2)
X2_scaled = scaler.fit_transform(X2)

print(f'스케일링 된 각 변수의 평균 : {X2_scaled[:,0].mean()}, {X2_scaled[:,1].mean()}, {X2_scaled[:,2].mean()}')
print(f'스케일링 된 각 변수의 표준편차 :{X2_scaled[:,0].std()}, {X2_scaled[:,1].std()}, {X2_scaled[:,2].std()}')
```

```
스케일링 된 각 변수의 평균 : 1.2212453270876723e-16, -4.529709940470639e-16, 2.220446049250313e-16
스케일링 된 각 변수의 표준편차 :1.0, 1.0, 0.9999999999999999
```

Linear Regression - 다중회귀

- 훈련세트/테스트세트 분할

```
# 훈련련세트, 테스트세트 분할
X_train, X_test, y_train, y_test  = train_test_split(X2_scaled, y, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(150, 2) (50, 2) (150,) (50,)
```

Linear Regression - 다중회귀

- 모델 생성 및 학습

```
# 모델 생성
model2 = LinearRegression()

# 모델 학습
model2.fit(X_train, y_train)
```

LinearRegression

LinearRegression()

Linear Regression - 다중회귀

- 모델 평가

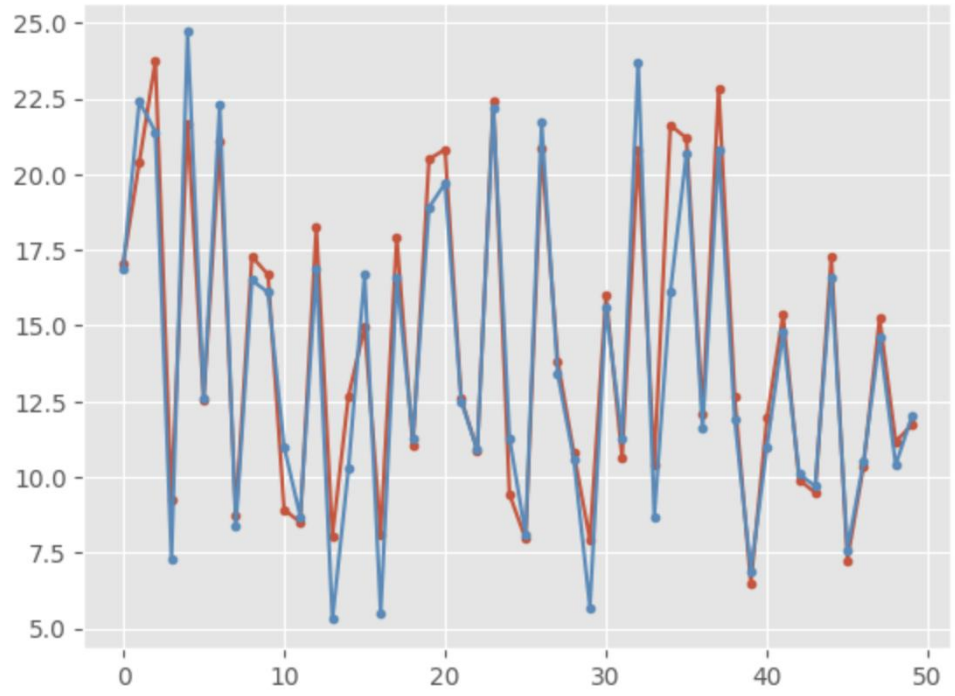
```
# 테스트데이터로 예측
pred = model2.predict(X_test)

# 모델 평가
mse = mean_squared_error(y_test, pred)
rmse = root_mean_squared_error(y_test, pred)
mae = mean_absolute_error(y_test, pred)
r2 = r2_score(y_test, pred)

print('===다중회귀 모델의 평가결과===')
print(f'MSE:{mse:.2f}')
print(f'RMSE:{rmse:.2f}')
print(f'MAE:{mae:.2f}')
print(f'R2:{r2:.2f}')
```

```
===다중회귀 모델의 평가결과===
MSE:2.40
RMSE:1.55
MAE:1.12
R2:0.91
```

Linear Regression - 다중회귀

- 모델 평가

```
plt.plot(pred, marker='.', label='pred')
plt.plot(np.array(y_test), marker='.', label='pred')
```



Linear Regression - 다중회귀

▪ 모델의 학습 결과

```
# 모델의 학습 결과
print(f'회귀계수:{model2.coef_}')
print(f'절편:{model2.intercept_}')

w1 = model2.coef_[0]
w2 = model2.coef_[1]
w3 = model2.coef_[2]

b = model2.intercept_
print(f'회귀식:{w1} * tv + {w2} * radio + {w3} * newspaper + {b}')
```

```
회귀계수:[4.67709227 1.4774023 0.09352121]
절편:15.22473459736965
회귀식:4.677092268978724 * tv + 1.4774023036363388 * radio + 0.0935212124232997 * newspaper + 15.22473459736965
```

Linear Regression - 다중회귀

▪ 새로운 값 예측하기

```
new_data = [[175,15,2]]
new_data_scaled = scaler.transform(new_data)
model2.predict(new_data_scaled)
```

```
array([15.80426509])
```

- 스케일링 된 데이터로 훈련했으므로, 예측할 데이터도 스케일링을 해주어야 한다.
- 훈련세트에서 사용한 스케일러를 이용하여 스케일링한다.
- 이 때는 fit 하지 않고 transform만 한다.

Linear Regression - 다중회귀

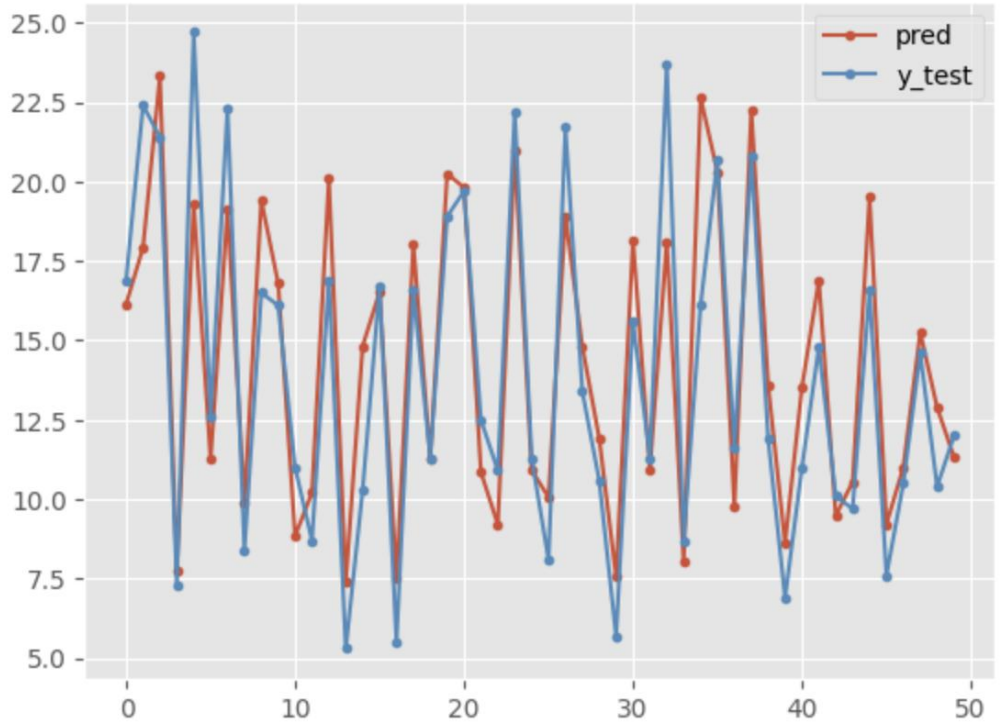
- 단순선형회귀 vs 다중회귀

단순선형회귀	다중회귀
MSE : 5.51	MSE:2.40
RMSE: 2.35	RMSE:1.55
MAE : 1.87	MAE:1.12
R2 : 0.80	R2:0.91

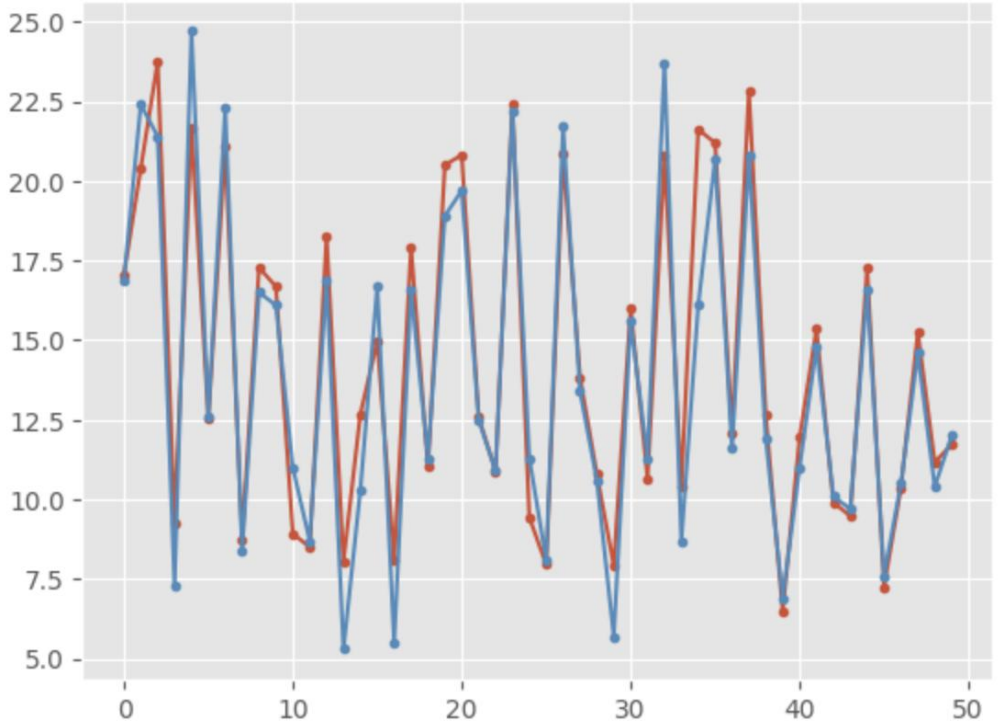
Linear Regression - 다중회귀

- 단순선형회귀 vs 다중회귀

단순선형회귀



다중회귀



Linear Regression – 다항회귀

독립변수를 2차, 3차방정식과 같은 다항식으로 표현하는 방법

세상의 모든 관계를 직선으로만 표현할 수는 없음

$$y = w1 * x1 + w2 * x2 + w1 * x1^2 + w2 * x2^2 + b$$

Linear Regression – 다항회귀

PolynomialFeatures

- scikit-learn 라이브러리에서 제공하는 클래스로, 주어진 특성(feature)들의 다항식 조합을 생성
- 원본 특성들의 조합을 통해 새로운 특성을 생성한다.
  - ✓ 입력 $[a, b]$가 주어졌을 때 degree=2인 경우 $[1, a, b, a^2, ab, b^2]$을 생성한다.
- 특성의 수가 크게 증가하여 과적합의 위험이 있을 수 있다.
- 다항식 특성은 원본 특성의 거듭제곱을 포함하므로, 특성 스케일링이 중요할 수 있다.

Linear Regression – 다항회귀

- 모델의 학습 결과

```
# 다항식 조합 생성
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X2)
print(f'특성의 크기 : {X_poly.shape}')

# 스케일링
scaler = StandardScaler()
X_poly_scaled = scaler.fit_transform(X_poly)

# 훈련련세트, 테스트세트 분할
X_train, X_test, y_train, y_test  = train_test_split(X_poly_scaled, y, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# 모델 생성 및 학습
model_poly = LinearRegression()
model_poly.fit(X_train, y_train)

# 평가
print(f'train score : {model_poly.score(X_train, y_train)}')
print(f'test score : {model_poly.score(X_test, y_test)}')
```

```
특성의 크기 : (200, 10)
(150, 10) (50, 10) (150,) (50,)
train score : 0.9276607239475623
test score : 0.9485618659991301
```

Linear Regression – 다항회귀

- 모델의 학습 결과

```
# 다항식 조합 생성
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=9)
X_poly = poly.fit_transform(X2)
print(f'특성의 크기 : {X_poly.shape}')

# 스케일링
scaler = StandardScaler()
X_poly_scaled = scaler.fit_transform(X_poly)

# 훈련련세트, 테스트세트 분할
X_train, X_test, y_train, y_test  = train_test_split(X_poly_scaled, y, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# 모델 생성 및 학습
model_poly = LinearRegression()
model_poly.fit(X_train, y_train)

# 평가
print(f'train score : {model_poly.score(X_train, y_train)}')
print(f'test score : {model_poly.score(X_test, y_test)}')
```

```
특성의 크기 : (200, 220)
(150, 10) (50, 10) (150,) (50,)
train score : 1.0
test score : -14619695.462740317
```

Ridge

▪ 규제

- 모델의 과적합을 방지하고 일반화 성능을 향상시키기 위해 사용되는 기법
- 모델의 복잡도를 제어하여 과적합을 방지한다.
  - 머신러닝 모델이 훈련세트를 너무 과도하게 학습하지 못하도록 훼방하여 과적합을 방지한다.
- 계수(가중치)에 패널티를 부여하는 방식으로 작동한다.

Ridge

▪ 규제

- **L2규제(Ridge 회귀)**
  - 모델의 **계수(가중치)의 제곱합**에 페널티를 부여하는 방식
  - 회귀 계수의 크기를 줄여 과도하게 큰 계수가 나타나는 것을 방지
  - **모든 계수를 작게 만들려고** 한다. 즉, 큰 값의 계수들이 줄어들지만, 완전히 0이 되지는 않는다.

- L1규제(Lasso 회귀)
  - 모델의 **계수(가중치)의 절대값 합**에 페널티를 부여하는 방식
  - 회귀 계수 중 불필요한 특성의 계수를 완전히 0으로 만들어서, **특성 선택**의 기능을 한다.

- Elastic Net
  - L1 규제와 L2 규제를 결합한 방식
  - 복잡한 데이터에서 좋은 성능을 낼 수 있다.

Ridge

- 규제

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=9)
X_ploy = poly.fit_transform(X2)
print(X_ploy.shape)

# 스케일링
scaler = StandardScaler()
X_poly_scaled = scaler.fit_transform(X_ploy)

# 훈련련세트, 테스트세트 분할
X_train, X_test, y_train, y_test  = train_test_split(X_poly_scaled, y, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# 모델 생성 및 학습
from sklearn.linear_model import Ridge
model_ridge = Ridge(alpha = 10) # alpha : 규제강도
model_ridge.fit(X_train, y_train)

# 평가
print(f'train score : {model_ridge.score(X_train, y_train)}')
print(f'train score : {model_ridge.score(X_test, y_test)}')
```

```
(200, 220)
(150, 220) (50, 220) (150,) (50,)
0.9216197291137791
0.944473790302263
```

공공자전거 수요예측

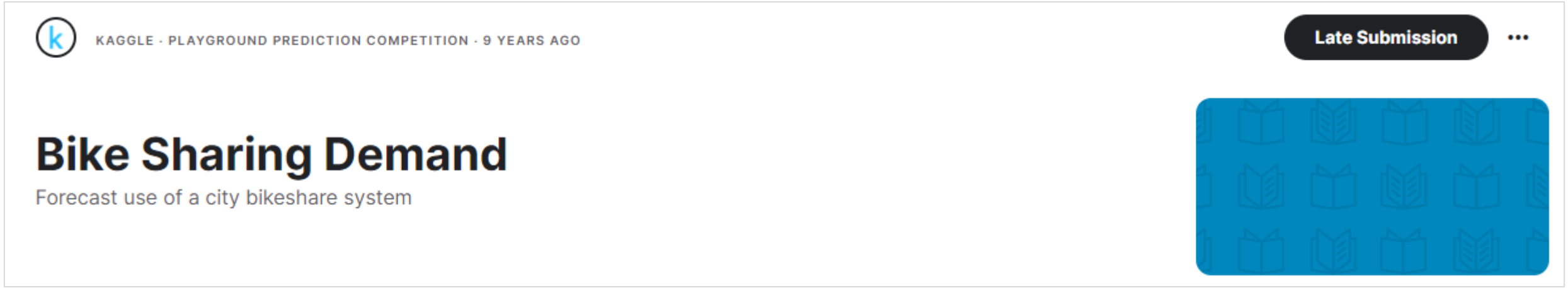
문제정의

목표 : 특정 시간의 공공자전거 대여량 예측(회귀)

목적 : 공공자전거의 적절한 배치와 관리

데이터 수집

https://www.kaggle.com/competitions/bike-sharing-demand/data

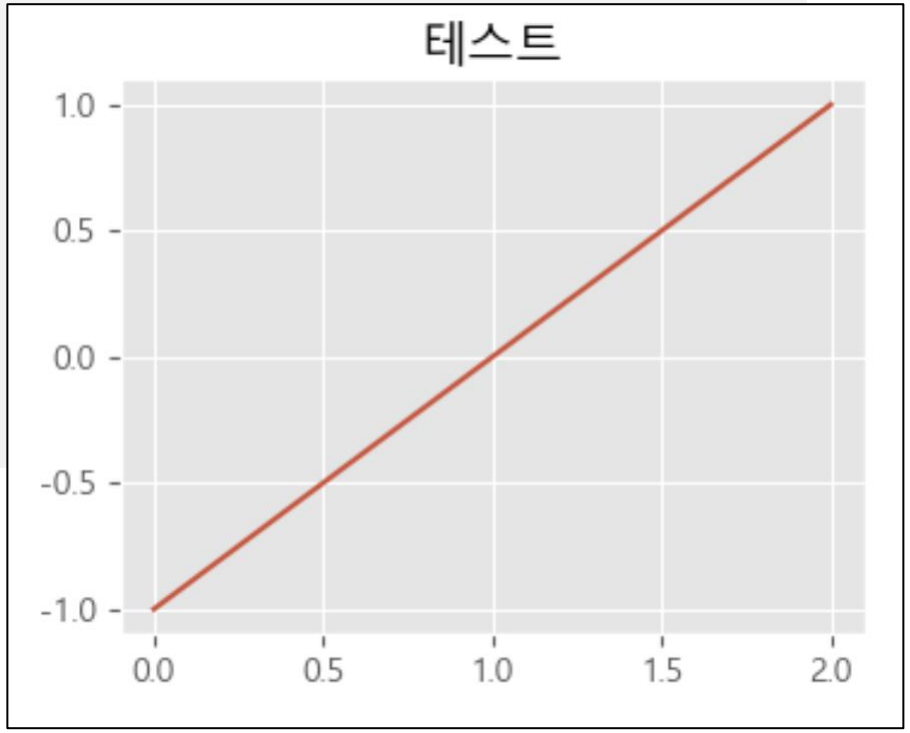


데이터 수집

- datetime - 시간대별 데이터 제공. timestamp(날짜와 시간 결합) 형태
- season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
- holiday - 공휴일(주말이 아닌 빨간날). 1-yes, 2-no
- workingday - 주말, 공휴일을 제외한 날 1-yes, 2-no
- weather
  - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp - 섭씨온도
- atemp - 섭씨체감온도
- humidity - 상대습도
- windspeed - 풍속
- casual - 비회원 대여량
- registered - 회원 대여량
- count - 총 대여량

작업준비

```
# 라이브러리 임포트
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')

# 한글폰트 설정
import matplotlib as mpl
mpl.rc('font', family='Malgun Gothic')
mpl.rcParams['axes.unicode_minus'] = False

pd.Series([-1,0,1]).plot(figsize=(4,3), title='테스트')
plt.show()
```



데이터 불러오기

```
# datetime 컬럼은 datetime 형식으로 읽어오도록 옵션 설정
df = pd.read_csv('data/bike_sharing_demand.csv', parse_dates=['datetime'])
display(df.head(3))
display(df.tail(3))
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

데이터 불러오기

```
# 데이터 정보
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

null 없음

문자열 없음

파생컬럼 추가

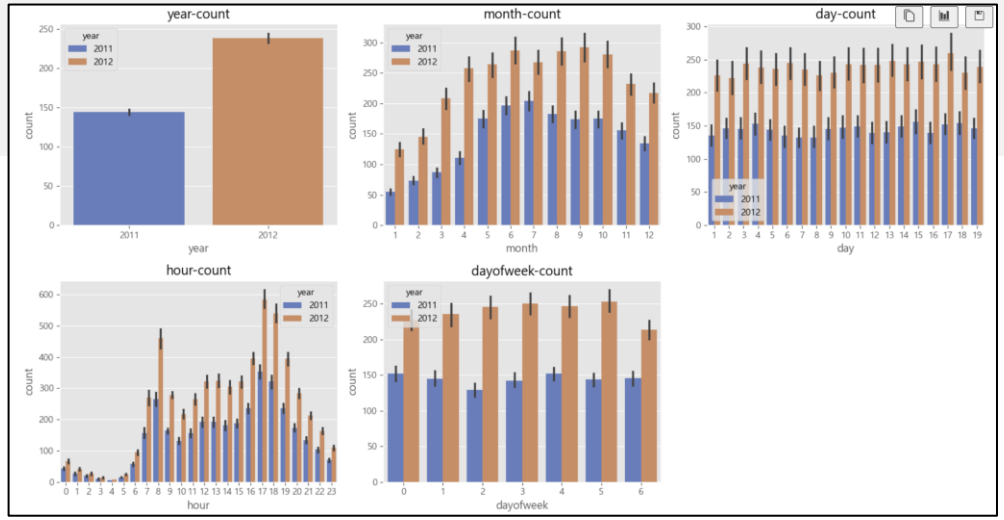
```
# 연, 월, 일, 시, 요일
df['year'] = df['datetime'].dt.year
df['month'] = df['datetime'].dt.month
df['day'] = df['datetime'].dt.day
df['hour'] = df['datetime'].dt.hour
df['dayofweek'] = df['datetime'].dt.dayofweek # 월요일:0, 일요일:6
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	year	month	day	hour	dayofweek
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011	1	1	0	5
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011	1	1	1	5
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011	1	1	2	5
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011	1	1	3	5
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011	1	1	4	5

데이터 탐색(EDA)

- datetime

```
plt.figure(figsize=(15,8))

features = ['year','month','day','hour','dayofweek']

for i, feature in enumerate(features):
    plt.subplot(2,3,i+1)
    sns.barplot(data=df, x=feature, y='count', estimator='mean', hue='year', palette='muted')
    plt.title(feature+'-count')

plt.tight_layout()
```

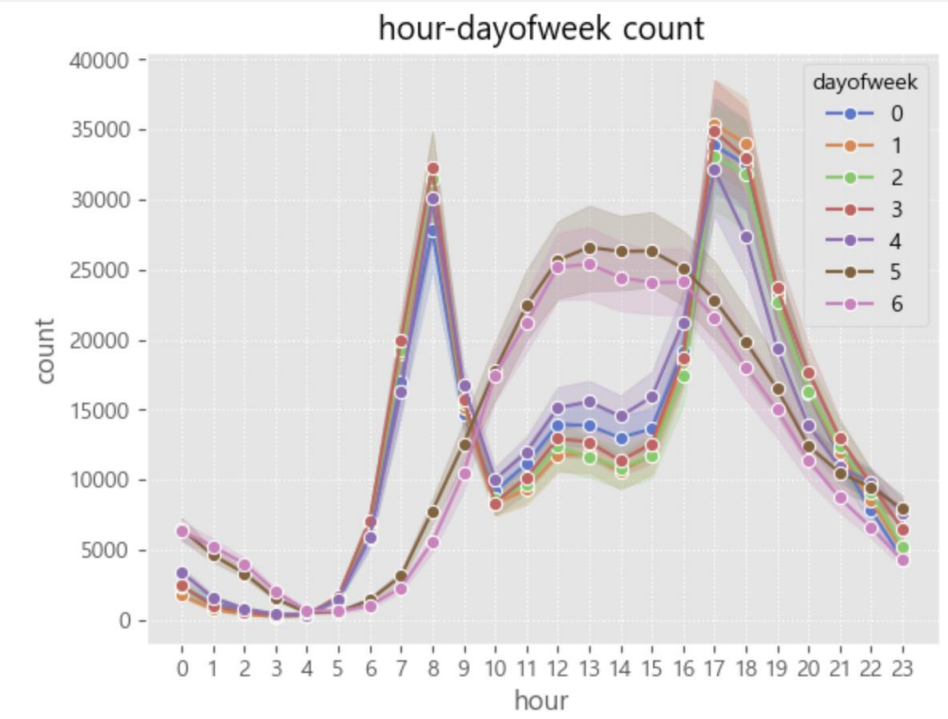


각 변수가
자전거 대여량와 어떤 연관성이 있는지 파악

데이터 탐색(EDA)

- datetime

```
# 시간대 - 요일 별 대여량
sns.lineplot(data=df, x='hour', y='count', marker='o', hue='dayofweek', estimator='sum', palette='muted')
plt.title('hour-dayofweek count')
plt.xticks(range(0,24))
plt.grid(ls=':')
plt.show()
```



시간대-요일이
자전거 대여량과 어떤 연관이 있는지 파악

데이터 탐색(EDA)

- season

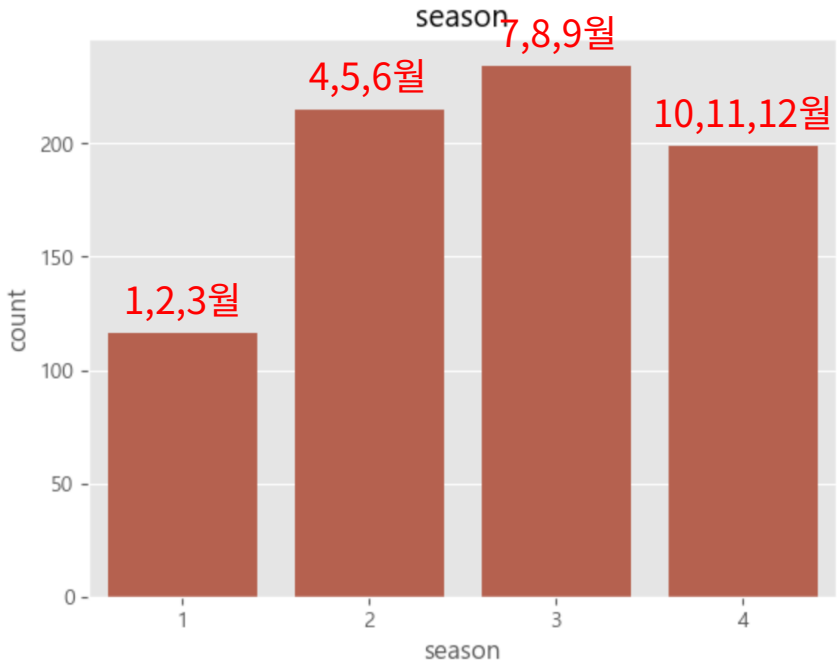
```
df[['month', 'season']].drop_duplicates()
```

	month	season
0	1	1
431	2	1
877	3	1
1323	4	2
1778	5	2
2234	6	2
2690	7	3
3146	8	3
3602	9	3
4055	10	4
4510	11	4
4966	12	4

데이터 탐색(EDA)

- season

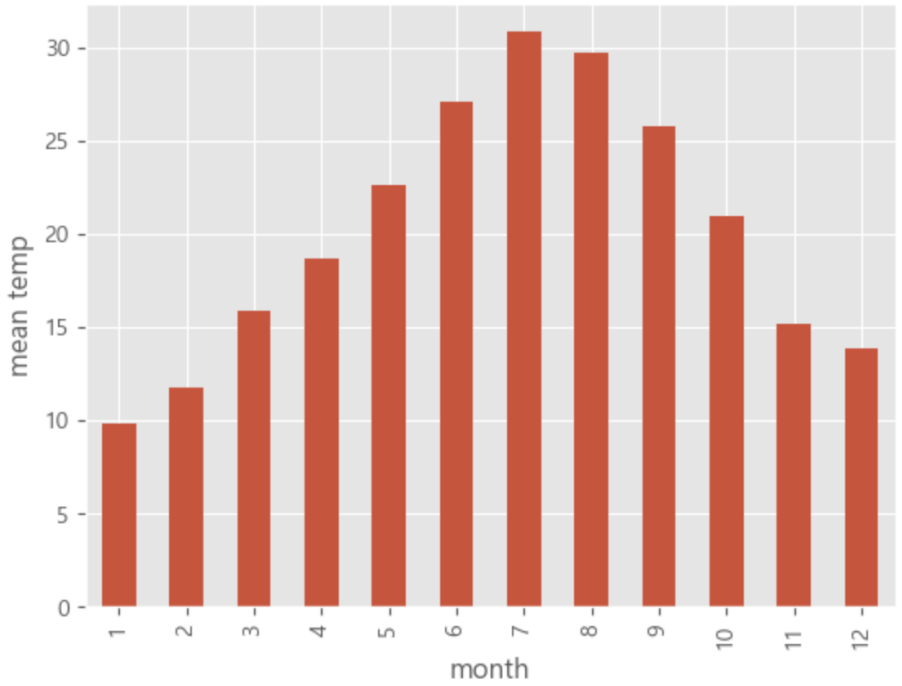
```
sns.barplot(data=df, x='season', y='count', errorbar=None)
plt.title('season')
plt.show()
```



데이터 탐색(EDA)

- season

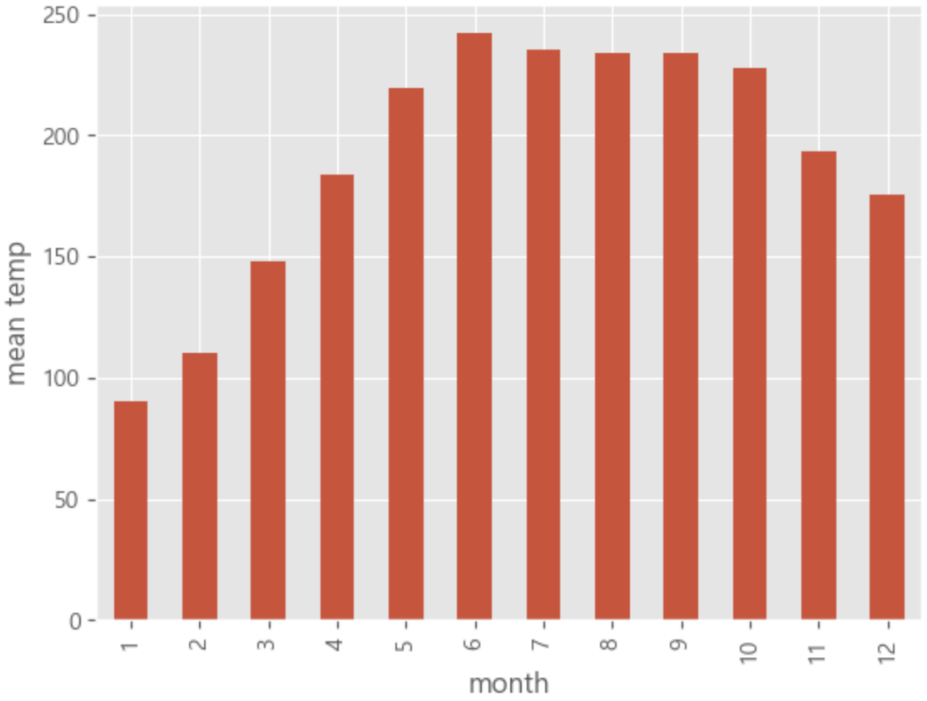
```
# 월별 평균 기온
df.groupby('month')['temp'].mean().plot.bar(ylabel='mean temp')
```



데이터 탐색(EDA)

- season

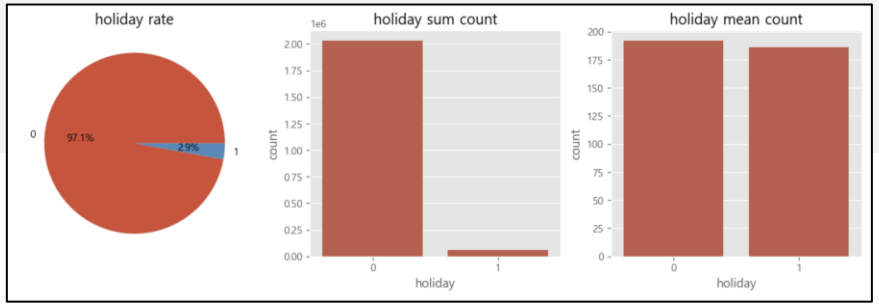
```
# 월별 평균 대여량
df.groupby('month')['count'].mean().plot.bar(ylabel='mean temp')
```



데이터 탐색(EDA)

- holiday

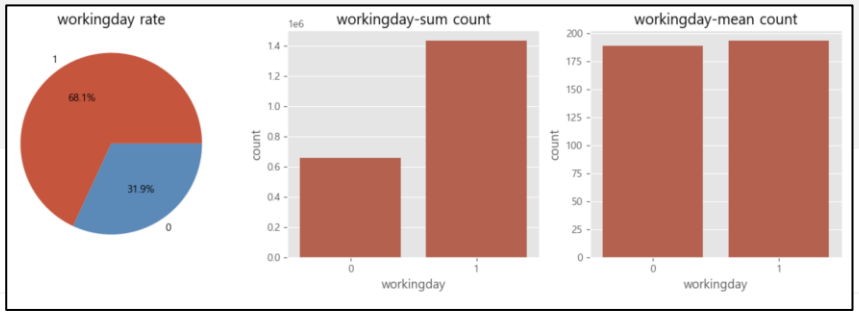
```
plt.figure(figsize=(15,4))

plt.subplot(131)
plt.pie(df['holiday'].value_counts(), autopct='%.1f%%', labels=['0','1'])
plt.title('holiday rate')

plt.subplot(132)
sns.barplot(data=df, x='holiday', y='count', errorbar=None, estimator='sum')
plt.title('holiday sum count')

plt.subplot(133)
sns.barplot(data=df, x='holiday', y='count', errorbar=None, estimator='mean')
plt.title('holiday mean count')

plt.show()
```



데이터 탐색(EDA)

- workingday

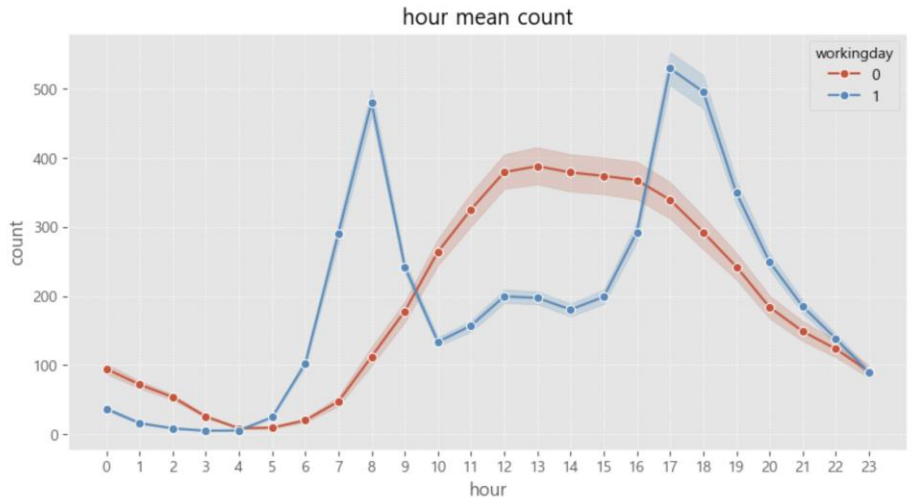
```
plt.figure(figsize=(15,4))

plt.subplot(131)
plt.pie(df['workingday'].value_counts(), autopct='%.1f%%', labels=['1','0'])
plt.title('workingday rate')

plt.subplot(132)
sns.barplot(data=df, x='workingday', y='count', errorbar=None, estimator='sum')
plt.title('workingday-sum count')

plt.subplot(133)
sns.barplot(data=df, x='workingday', y='count', errorbar=None, estimator='mean')
plt.title('workingday-mean count')

plt.show()
```



데이터 탐색(EDA)

- workingday

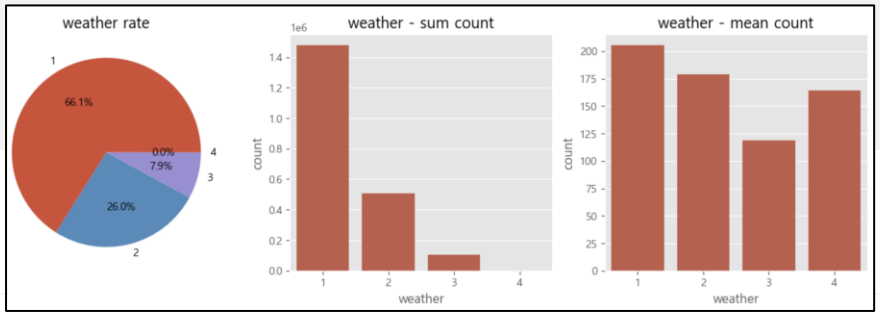
```
plt.figure(figsize=(10,5))
sns.lineplot(data=df, x='hour', y='count', marker='o', hue='workingday')
plt.title('hour mean count')
plt.xticks(range(0,24))
plt.grid(ls=':')
plt.show()
```



데이터 탐색(EDA)

- weather

```
plt.figure(figsize=(15,4))

plt.subplot(131)
plt.pie(df['weather'].value_counts(), autopct='%.1f%%', labels=[1,2,3,4])
plt.title('weather rate')

plt.subplot(132)
sns.barplot(data=df, x='weather', y='count', errorbar=None, estimator='sum')
plt.title('weather - sum count')

plt.subplot(133)
sns.barplot(data=df, x='weather', y='count', errorbar=None, estimator='mean')
plt.title('weather - mean count')

plt.show()
```



데이터 탐색(EDA)

- weather

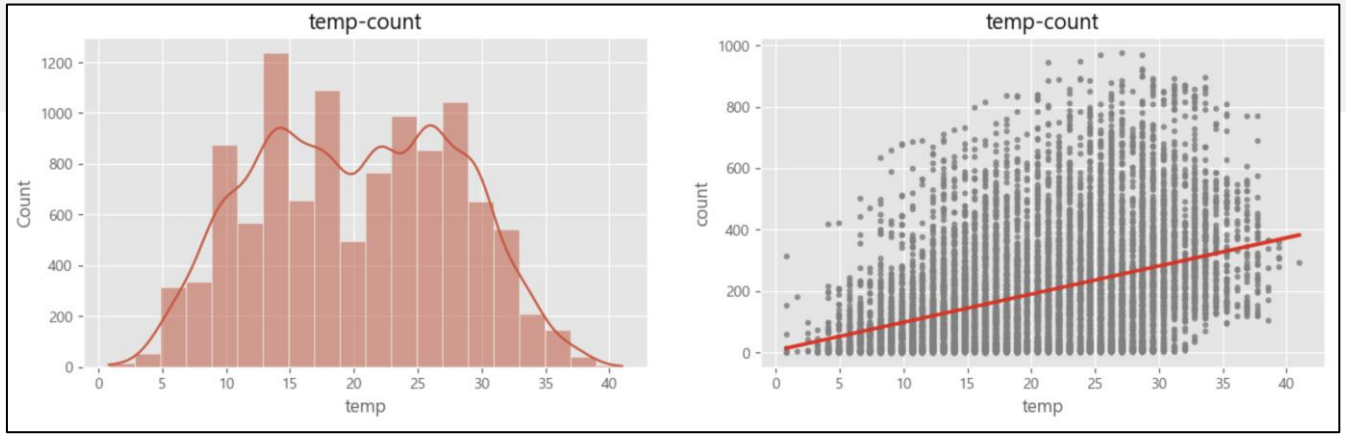
```
df['weather'].value_counts()
```

```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```

데이터 탐색(EDA)

- temp

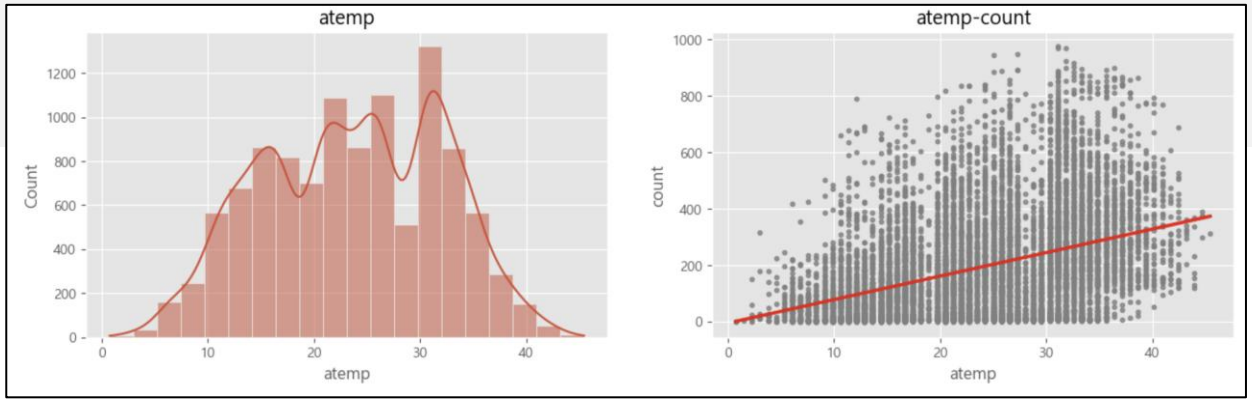
```
plt.figure(figsize=(15,4))

plt.subplot(121)
sns.histplot(df['temp'], bins=20, kde=True)
plt.title('temp-count')

plt.subplot(122)
sns.regplot(data=df, x='temp', y='count', color='gray', line_kws={'color':'red'}, marker='.')
plt.title('temp-count')

plt.show()
```



데이터 탐색(EDA)

- atemp

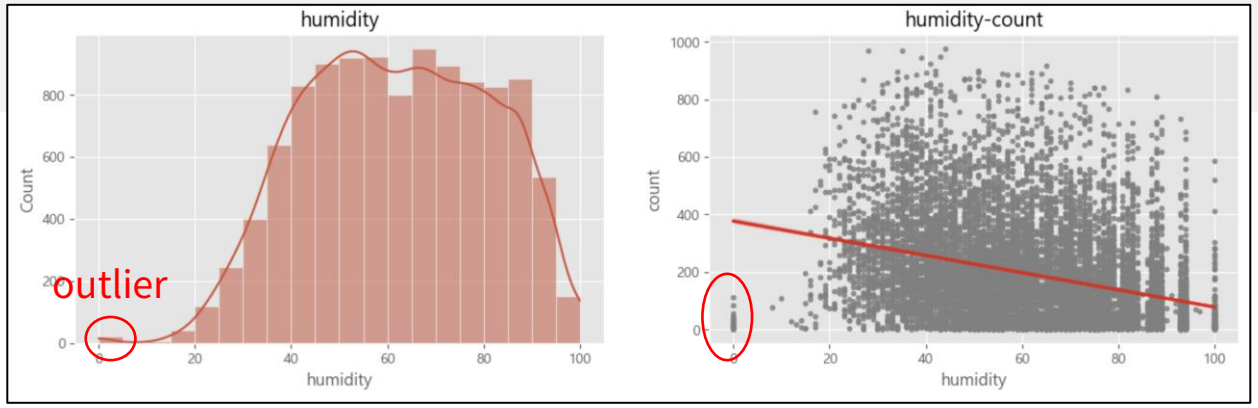
```
plt.figure(figsize=(15,4))

plt.subplot(121)
sns.histplot(df['atemp'], bins=20, kde=True)
plt.title('atemp')


plt.subplot(122)
sns.regplot(data=df, x='atemp', y='count', color='gray', line_kws={'color':'red'}, marker='.')
plt.title('atemp-count')

plt.show()
```



데이터 탐색(EDA)

- humidity

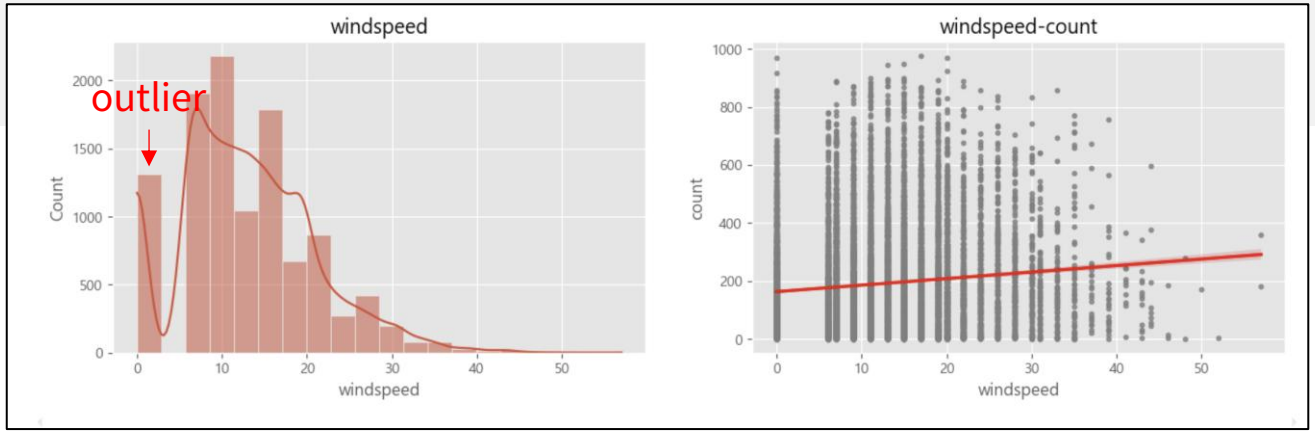
```
plt.figure(figsize=(15,4))

plt.subplot(121)
sns.histplot(df['humidity'], bins=20, kde=True)
plt.title('humidity')

plt.subplot(122)
sns.regplot(data=df, x='humidity', y='count', color='gray', line_kws={'color':'red'}, marker='.')
plt.title('humidity-count')

plt.show()
```



데이터 탐색(EDA)

- windspeed

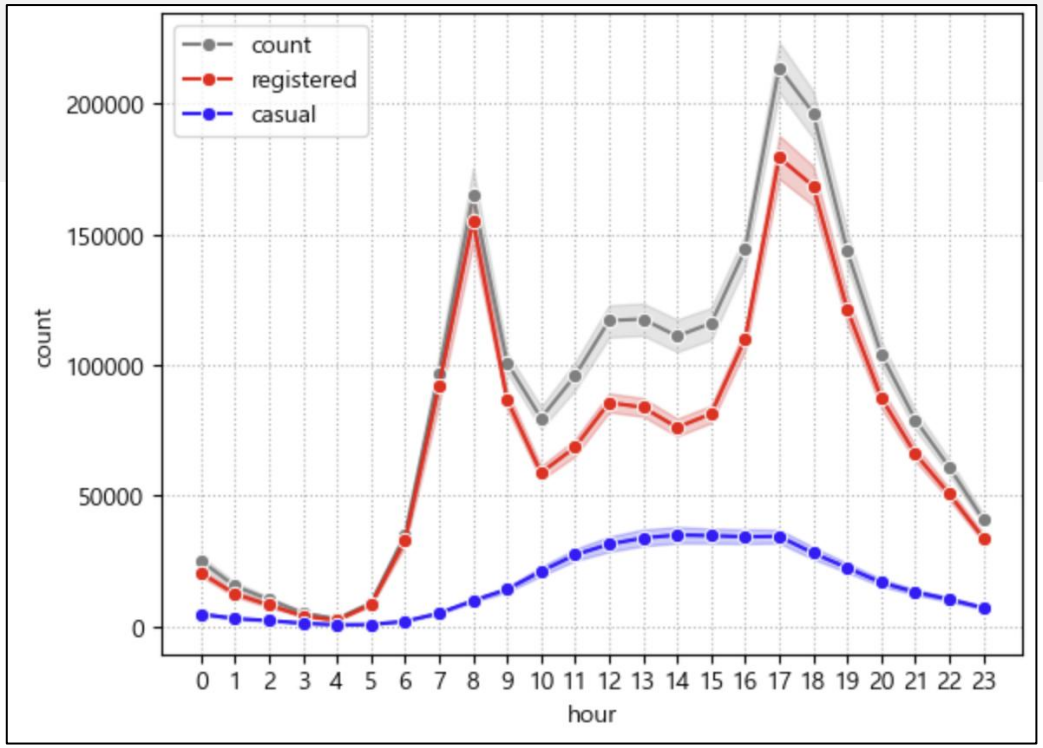
```
plt.figure(figsize=(15,4))

plt.subplot(121)
sns.histplot(df['windspeed'], bins=20, kde=True)
plt.title('windspeed')

plt.subplot(122)
sns.regplot(data=df, x='windspeed', y='count', color='gray', line_kws={'color':'red'}, marker='.')
plt.title('windspeed-count')

plt.show()
```



데이터 탐색(EDA)

- casual, registered, count

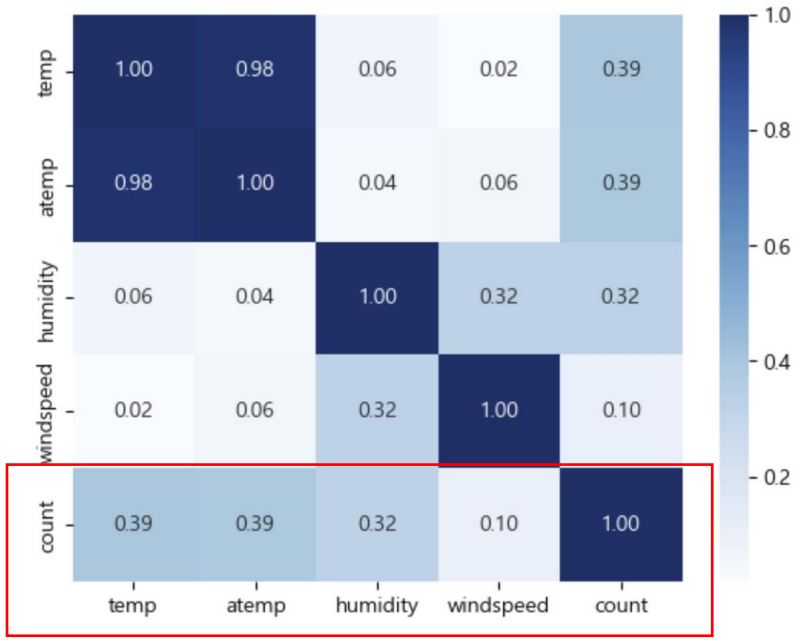
```
sns.lineplot(data=df, x='hour', y='count', color='gray', marker='o', estimator='sum', label='count')
sns.lineplot(data=df, x='hour', y='registered', color='red', marker='o', estimator='sum', label='registered')
sns.lineplot(data=df, x='hour', y='casual', color='blue', marker='o', estimator='sum', label='casual')
plt.legend()
plt.xticks(range(0,24))
plt.grid(ls=':')
plt.show()
```



데이터 탐색(EDA)

▪ 상관분석

```
corr_matt = ['temp','atemp','humidity','windspeed','count']
corr = abs(df[corr_matt].corr())
sns.heatmap(corr, annot=True, cmap='Blues', fmt='.2f')
```



머신러닝

▪ 변수선택

```
# 변수선택
'''
- casual, registered는 독립변수의 성격이 아니므로 선택하지 않음
- datetime은 선택하지 않음
'''
X = df.drop(['datetime','casual','registered','count'], axis=1).copy()
y = df['count']
```

머신러닝

- 훈련세트, 테스트세트 분할

```
# 훈련세트/테스트세트 분할
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(8164, 13) (2722, 13) (8164,) (2722,)
```

머신러닝

▪ 모델 생성 및 훈련

```
# 모델 생성
from sklearn.linear_model import LinearRegression
model = LinearRegression()  # 모델 생성
model.fit(X_train, y_train) # 모델 훈련
```

```
LinearRegression()
```

머신러닝

▪ 모델 성능 평가

```
# 테스트데이터로 예측
pred = model.predict(X_test)

# 모델 성능 평가
from sklearn.metrics import root_mean_squared_error, r2_score
rmse = root_mean_squared_error(y_test, pred)
r2 = r2_score(y_test, pred)

print(f'rmse:{rmse}')
print(f'r2:{r2}')
```

```
rmse:141.74025022251396
r2:0.38775297175891066
```

머신러닝

▪ 원핫인코딩

```
# 변수선택
'''
- temp와 atemp는 다중공선성이 강하게 존재하므로 둘중 하나만 선택
- casual, registered는 독립변수의 성격이 아니므로 선택하지 않음
- datetime은 선택하지 않음
'''
X = df.drop(['datetime','temp','casual','registered','count'], axis=1).copy()
y = df['count']

# 범주형변수 원핫인코딩
X_ohe = pd.get_dummies(X, columns=['year','month','hour','dayofweek','day','season','weather’])

display(X_ohe.head())
print(X_ohe.shape)
```

	holiday	workingday	atemp	humidity	windspeed	year_2011	year_2012	month_1	month_2	month_3	...	day_18	day_19	season_1	season_2	season_3	season_4	weather_1	weather_2	weather_3	weather_4
0	0	0	14.395	81	0.0	True	False	True	False	False	...	False	False	True	False	False	False	True	False	False	False
1	0	0	13.635	80	0.0	True	False	True	False	False	...	False	False	True	False	False	False	True	False	False	False
2	0	0	13.635	80	0.0	True	False	True	False	False	...	False	False	True	False	False	False	True	False	False	False
3	0	0	14.395	75	0.0	True	False	True	False	False	...	False	False	True	False	False	False	True	False	False	False
4	0	0	14.395	75	0.0	True	False	True	False	False	...	False	False	True	False	False	False	True	False	False	False

```
(10886, 77)
```

머신러닝

- 원핫인코딩 된 데이터로 모델 생성 및 평가

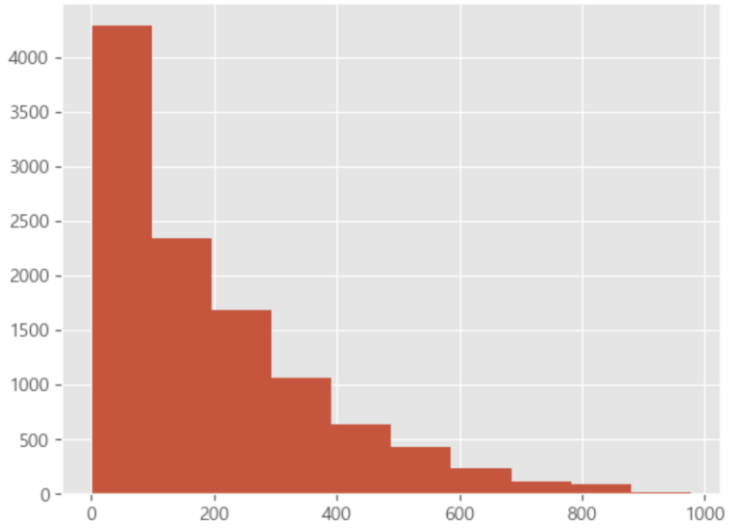
```
# 훈련세트/테스트세트 변환
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_ohe, y, random_state=42)

# 모델 생성
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)

# 모델 성능 평가
pred = model.predict(X_test)
from sklearn.metrics import root_mean_squared_error, r2_score
rmse = root_mean_squared_error(y_test, pred)
r2 = r2_score(y_test, pred)

print(f'rmse:{rmse}')
print(f'r2:{r2}')
```

```
rmse:100.55563444113648
r2:0.691856425225667
```

머신러닝

- 종속변수 로그변환

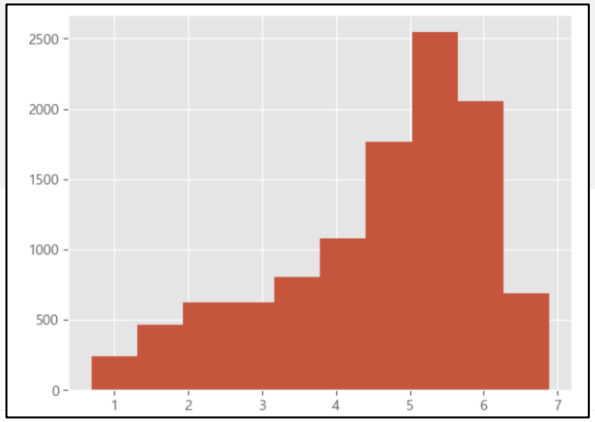
```
y.hist()
```



머신러닝

▪ 종속변수 로그변환

```
# 변수선택
'''
- temp와 atemp는 다중공선성이 강하게 존재하므로 둘중 하나만 선택
- casual, registered는 독립변수의 성격이 아니므로 선택하지 않음
- datetime은 선택하지 않음
'''
X = df.drop(['datetime','temp','casual','registered','count'], axis=1).copy()
y = df['count']

# 범주형변수 원핫인코딩
X_ohe = pd.get_dummies(X, columns=['year','month','hour','dayofweek','day','season','weather'])

# 종속변수 로그변환
y_log = np.log1p(y)
y_log.hist()
```



머신러닝

▪ 종속변수 로그변환

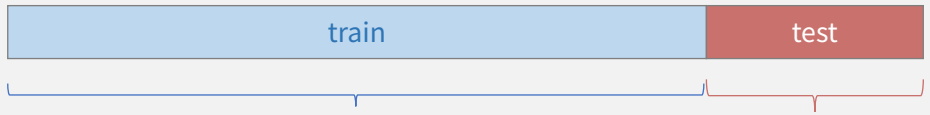
```
# 훈련세트/테스트세트 변환
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_ohe, y_log, random_state=42)

# 모델 생성
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)

# 모델 성능 평가
pred = model.predict(X_test)

# 로그 역변환
pred_original = np.expm1(pred)
y_test_original = np.expm1(y_test)

from sklearn.metrics import root_mean_squared_error, r2_score
rmse = root_mean_squared_error(y_test_original, pred_original)
r2 = r2_score(y_test_original, pred_original)

print(f'rmse:{rmse}')
print(f'r2:{r2}')
```

```
rmse:95.83993264536748
r2:0.7200804087915887
```

모델의 평가 방법

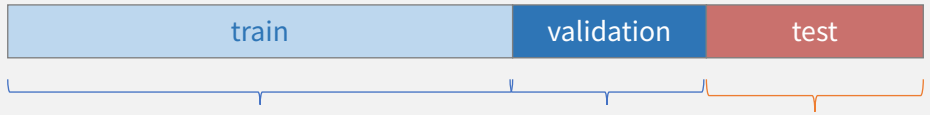
홀드아웃 검증
Holdout method

전체 데이터를 훈련용/테스트용으로
또는 훈련용/검증용/테스트용으로 분할



훈련

평가
- 과적합 확인
- 하이퍼파라미터 최적화



훈련

중간평가
- 과적합 확인
- 하이퍼파라미터 최적화

최종 평가

k-폴드 교차검증
k-Fold Cross-Validation

학습 데이터를 k개의 동일한 크기의 부분집합으로 나누고,
k-1개의 집단을 학습용으로, 나머지는 검증용으로 설정하여 학습
k번 반복 측정한 결과를 평균 낸 값을 검증 결과값으로 사용

머신러닝

- 교차검증

```
model = LinearRegression()

from sklearn.model_selection import cross_val_score, KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)
print(cross_val_score(model, X, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y_log, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X, y_log, scoring='r2', cv=kf ).mean())
```

```
0.3877043694112919
0.6899632708700877
0.8305572927459458
0.48576195811118456
```

머신러닝

- 다른 모델과 비교 – K최근접이웃

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor()

from sklearn.model_selection import cross_val_score, KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)
print(cross_val_score(model, X, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y_log, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X, y_log, scoring='r2', cv=kf ).mean())
```

```
0.5479108832500819
0.2852235953854074
0.279639695700112
0.6563442821767372
```

머신러닝

- 다른 모델과 비교 – 의사결정트리

```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()

from sklearn.model_selection import cross_val_score, KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)
print(cross_val_score(model, X, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y_log, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X, y_log, scoring='r2', cv=kf ).mean())
```

```
0.8948815637507694
0.8450813117449174
0.8967760154002591
0.9061500546093708
```

머신러닝

- 다른 모델과 비교 – 랜덤포레스트

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()

from sklearn.model_selection import cross_val_score, KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)
print(cross_val_score(model, X, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X_ohe, y_log, scoring='r2', cv=kf ).mean())
print(cross_val_score(model, X, y_log, scoring='r2', cv=kf ).mean())
```

```
0.9513199662054964
0.922004151449662
0.946128358240051
0.9544268758958647
```

모듈 및 함수 정리

구분	모듈	함수/클래스
데이터 전처리	sklearn.preporcessing	StandardScaler 데이터 스케일링(표준화 : 평균0, 표준편차1) MinMaxScaler 데이터 스케일링(정규화 : 최소0, 최대1) RobustScaler 데이터 스케일링(중위수, IQR 사용)
데이터 분리 및 검증	sklearn.model_selection	train_test_split 훈련/테스트 데이터 분할 KFold 교차검증을 위한 데이터 분할 cross_val_scores 교차검증
머신러닝	sklearn.linear_model	LinearRegression 선형회귀 Ridge 릿지회귀(규제)
성능 평가	sklearn.metrics	mean_squared_error (mse) root_mean_squared_error (rmse) mean_absolute_error (mae) r2_score (r2)

메소드/프로퍼티 정리

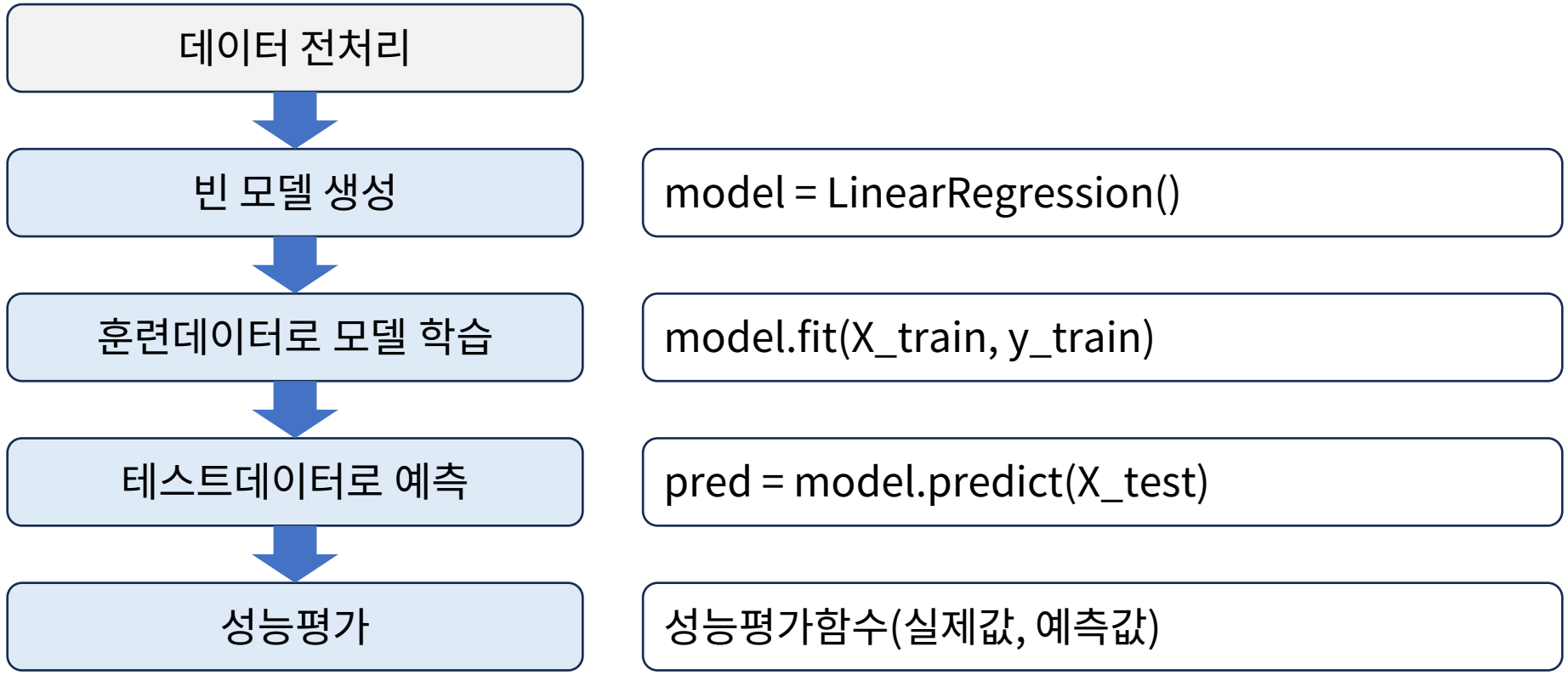
예측기(선형회귀)

- model = LinearRegression() : 모델 생성
- model.fit(X_train, y_train) : 훈련
- model.predict(X_test) : 예측
- model.score(X, y) : r2_score
- model.coef_ : 휘귀계수
- model.intercept_ : 절편

변환기(표준화)

- scaler = StandardScaler() : 변환기 생성
- scaler.fit(X_train, y_train) : 학습
- scaler.transform(X_test) : 변환
- scaler.fit_transform(X, y) : 학습 및 변환

모델생성 및 평가 절차



모델생성 및 평가 절차