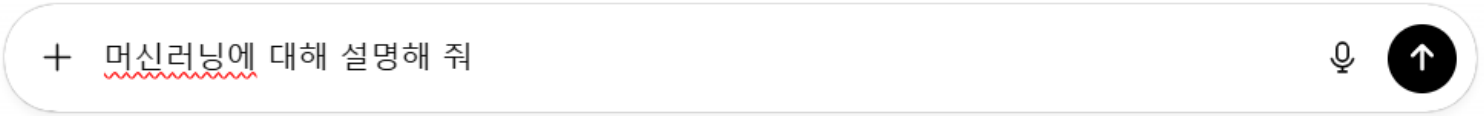
13장. 머신러닝 및 딥러닝

머신러닝(Machine Learning)



머신러닝이란? 명시적으로 규칙을 코딩하지 않아도, 데이터로부터 패턴을 학습해 예측·판단하는 인공지능 기술

머신러닝의 핵심 개념

1 데이터

- 학습의 재료
- 많고, 정확할수록 성능이 좋아짐

2 모델(Model)

- 데이터를 학습한 결과물
- 예: "이메일이 스팸일 확률은 92%"

3 학습(Training)

- 데이터를 보고 모델이 규칙을 조정하는 과정

4 예측(Prediction)

- 새로운 데이터에 대해 결과를 맞혀보는 것

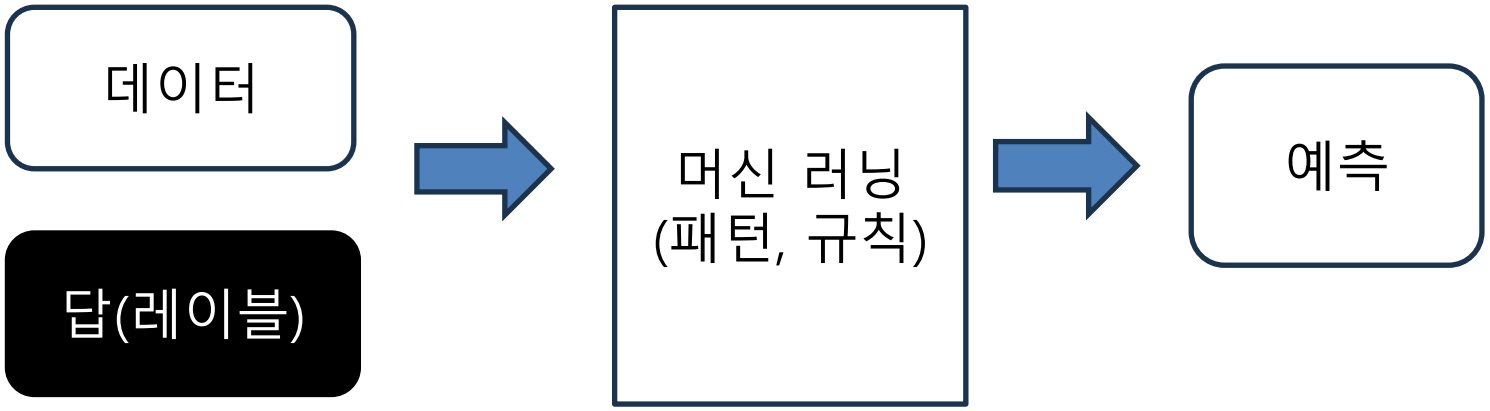
머신러닝(Machine Learning)

❖ **지도 학습 vs 비지도 학습**

	지도 학습	비지도학습
분석 모형 (알고리즘)	• 회귀 • 분류	• 군집 • 차원 축소
특징	• 정답(레이블)을 알고 있는 상태에서 학습 • 회귀 – 가격, 매출, 주가 예측 • 분류 – 고객분류, 질병 진단, 스팸 메일 필터링	• 정답(레이블)이 없는 상태에서 서로 비슷한 데이터를 찾아서 그룹화(클러스터링) • 구매 패턴 분석, 유투브 추천 등

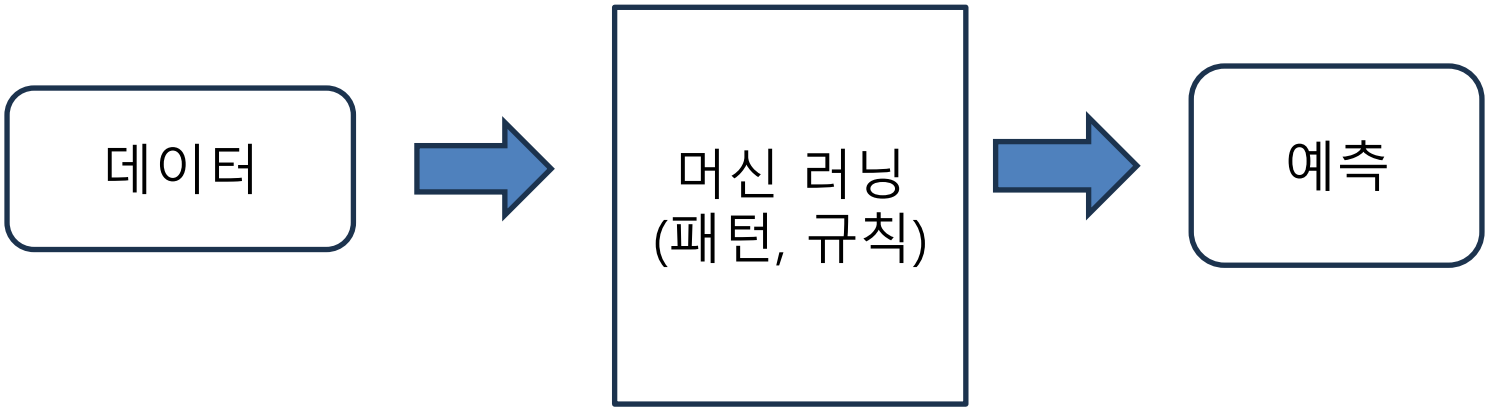
머신러닝(Machine Learning)

❖ **지도 학습**



머신러닝(Machine Learning)

❖ **비지도 학습**



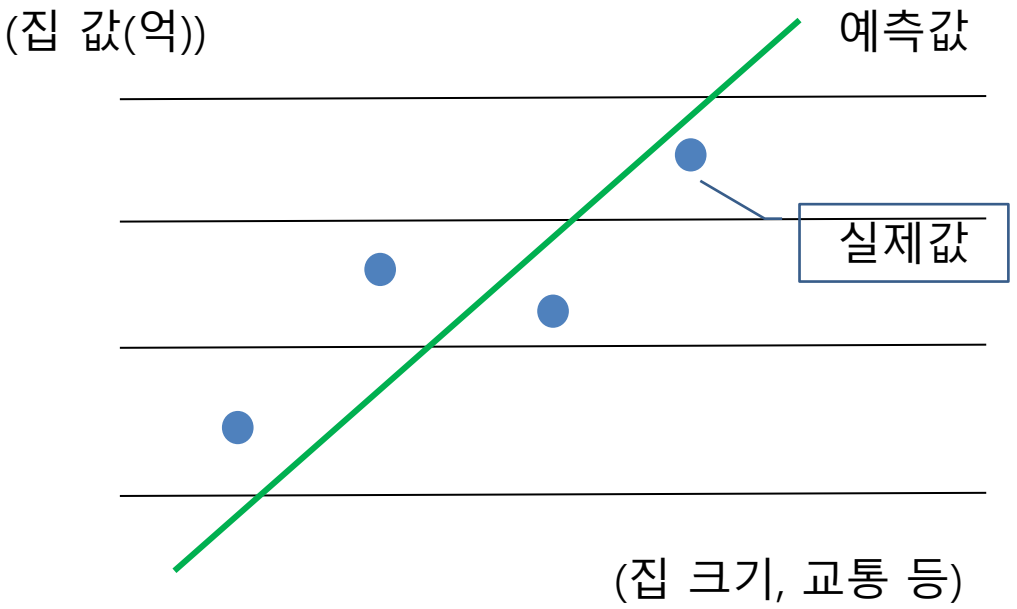
머신러닝(Machine Learning)

❖ 강화 학습

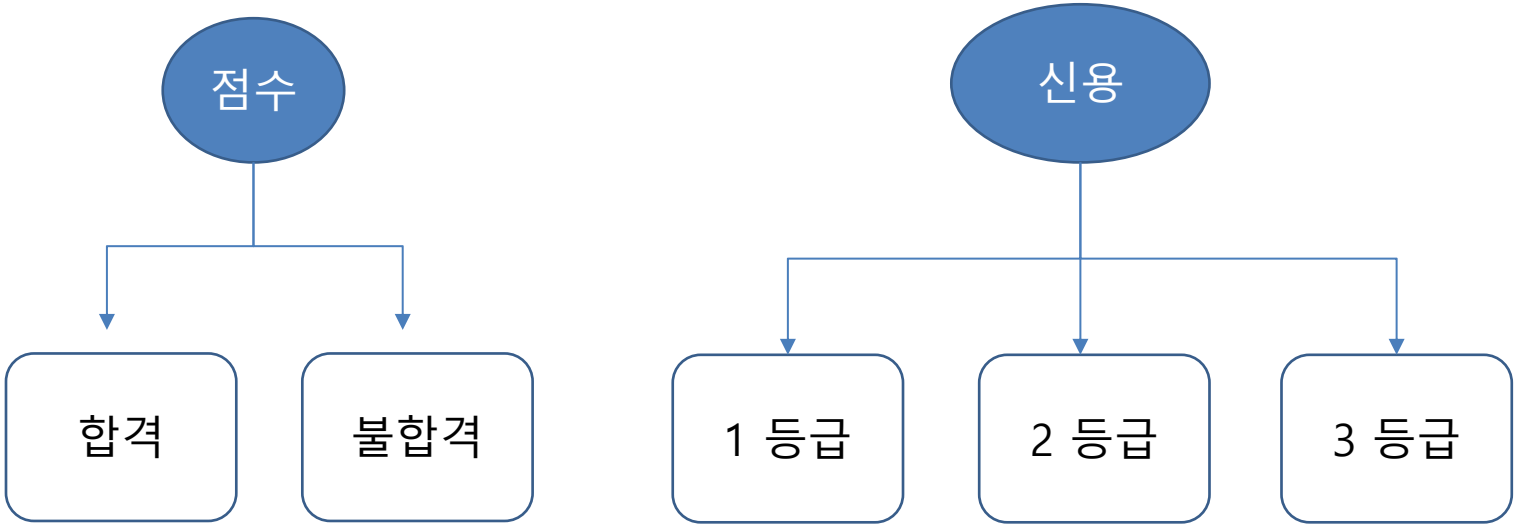
- 강화란 '어떤 것의 수준이나 정도를 높인다'는 의미로 인공지능의 수준을 높이는 것을 말한다.
- 에이전트가 환경과 상호작용하면서 보상 신호에 따라 최적의 정책을 찾아내는 방법

- 벽돌깨기 게임 – 딥마인드(구글) 회사
- 알파고(딥마인드) – 바둑을 학습한 인공지능
- 자율주행 자동차, 인공지능 로봇으로 확장되고 있음

회귀(Regression)

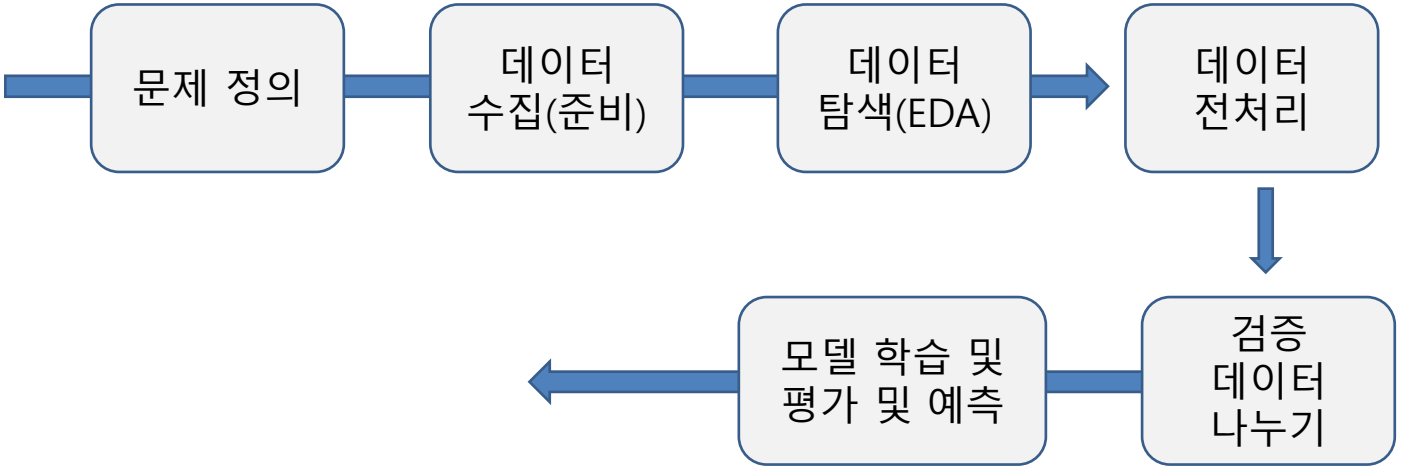


분류(Classification)



머신러닝 프로세스

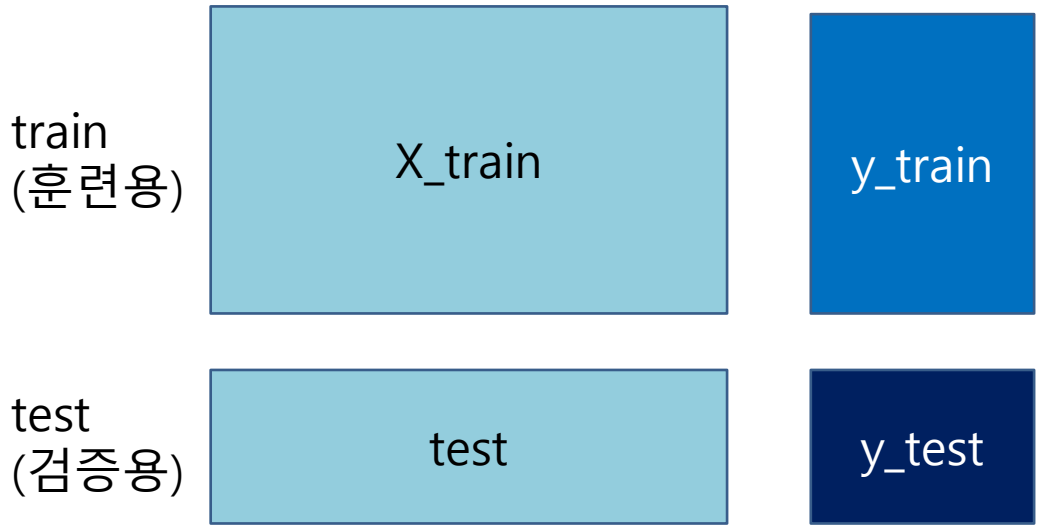
- 머신러닝 모델 예측 프로세싱



데이터 수집(준비)

- 데이터는 학습용과 검증으로 준비한다.
  - ✓ 훈련 : 검증 비율

    80 : 20 or 70 : 30



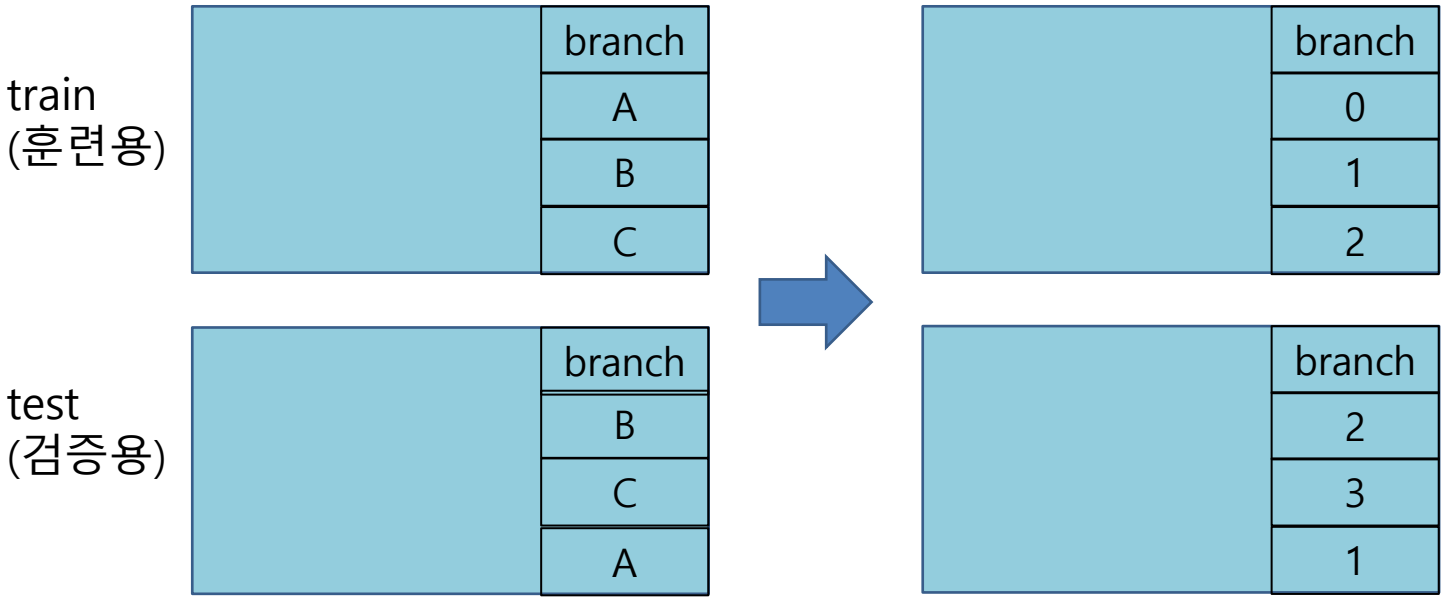
데이터 전처리

- 데이터 처리를 위한 모듈

	알고리즘	필수 여부
레이블 인코딩	sklearn.preprocessing(모듈) LabelEncoder	필수 (원-핫 인코딩중 선택)
스케일링	sklearn. preprocessing(모듈) StandardScaler	선택
데이터 분할	sklearn. model_selection(모듈) train_test_split	필수

데이터 전처리

- 인코딩(Encoding) – 문자형을 숫자형으로 변환
  - 레이블 인코딩(Label Encoding) : 각 칼럼을 숫자로 매핑하는 인코딩 방법이다.



데이터 전처리

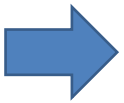
- 인코딩(Encoding) – 문자형을 숫자형으로 변환
  - 원-핫 인코딩(One-Hot Encoding) : 각 칼럼에 대해 새로운 칼럼을 만들고 새로운 칼럼에 0 또는 1이 저장된다.

train (훈련용)

branch
A
B
C

test (검증용)

branch
B
C
A



branch_A	branch_B	branch_C
1	0	0
0	1	0
0	0	1

branch_A	branch_B	branch_C
0	1	0
0	0	1
1	0	0

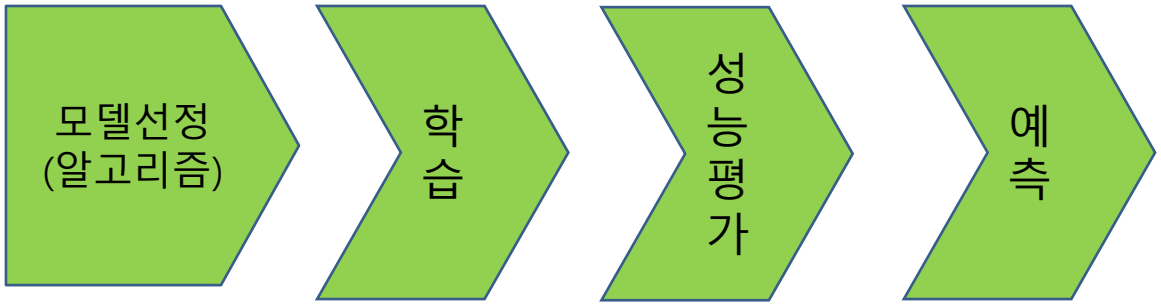
모델 학습 및 평가

- 모델 학습 및 평가

지도 학습	알고리즘	평가 지표
분류	sklearn.ensemble(모듈) RandomForestClassifier	sklearn.metrics(모듈) roc_auc_score, f1_score
회귀	sklearn.linear_model(모듈) LinearRegression sklearn.ensemble(모듈) RandomForestRegressor	sklearn.metrics(모듈) mean_squared_error mean_absolute_error

모델 학습 및 평가

- 모델 학습, 평가 및 예측



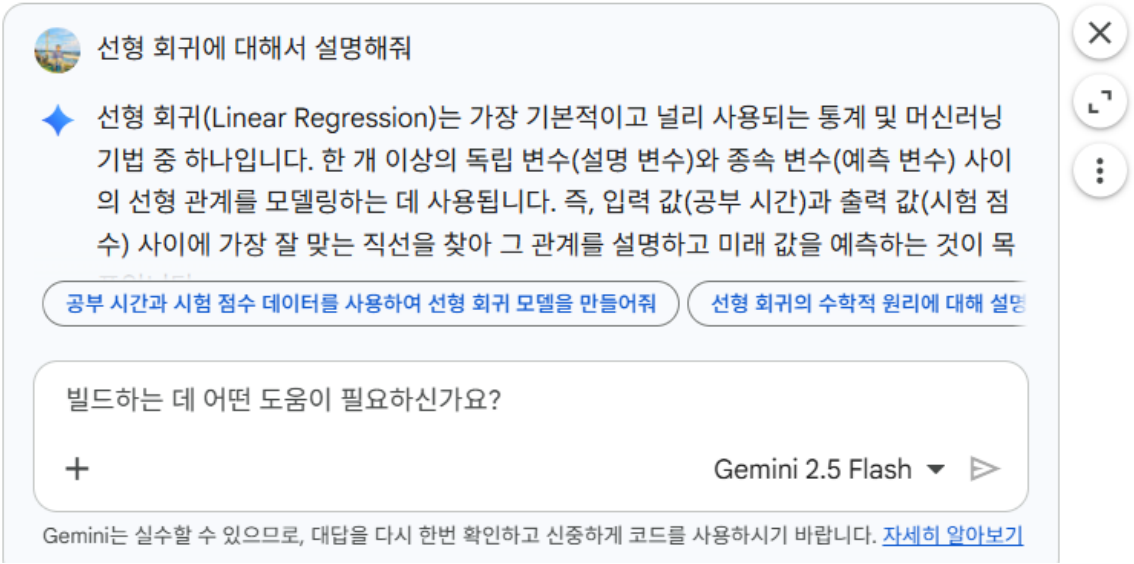
회귀(Regression)

- **상관 분석 vs 회귀 분석**

상관 분석	회귀 분석
x와 y의 관련 정도를 확인	x가 y에 주는 영향을 추정하여 결과를 예측할 수 있음
산포도, 상관계수	회귀식
광고비와 매출은 양의 상관 관계가 있다.	광고비를 1% 늘리면 매출이 5% 증가할 것이다.

회귀(Regression)

- **선형 회귀**



선형 회귀 분석

- **단순 선형 회귀**
  - $y=Wx+b$
  - W 를 가중치(Weight), b를 편향(bias)라고 부른다
  - 그래프의 형태는 **직선**

- **다중 선형 회귀**
  - $y=W_1x_1 + W_2x_2 + \ldots + W_nx_n + b$
  - 만약 2개의 독립 변수면 그래프는 **곡선**으로 나타남

선형 회귀 분석

- **공부한 시간의 차이에 따른 시험 성적 예측하기**
  - 성적을 변하게 하는 요소를 x라 하고 이 x값에 따라 변하는 '성적'을 y라 하자.
  - 독립 변수(x) : 공부한 시간, 사교육비 지출, 학생의 지능지수 등
  - 종속 변수(y) : 성적
  - 어떤 변수가 다른 변수에 영향을 준다면 두 변수 사이에 선형관계가 있다.

선형 회귀 분석

- **공부한 시간의 차이에 따른 시험 성적 예측하기**

공부 시간	2시간	4시간	6시간	8시간
성적	81점	93점	91점	97점

x = {2, 4, 6, 8}

y = {81, 93, 91, 97}

직선의 방정식을 구한다.

$y = ax + b$(일차함수)

기울기 a와 절편 b를 알아야 함

최소 제곱법

- **키의 차이에 따른 몸무게 예측하기**
  - 기울기 a 구하기

$$a = \frac{(x - x\text{평균})(y - y\text{평균})\text{의 합}}{(x - x\text{평균})^2\text{의 합}}$$

공부 시간(x) 평균 : (2 + 4 + 6 + 8) ÷ 4 = 5

성적(y) 평균: (81 + 93 + 91 + 97) ÷ 4 = 90.5

기울기 a = 2.3

X의 편차(각 값과 평균과의 차이)를 제곱해서 합한 값을 분모로 놓고,
x와 y의 편차를 곱해서 합한 값을 분자로 놓으면 기울기가 나옴

최소 제곱법

- **공부한 시간의 차이에 따른 시험 성적 예측하기**

  - 편차(deviation)

    관측값에서 평균을 뺀 값

  - 분산 (variance)

    편차(관측값-평균)를 제곱하고 그것을 모두 더한 후
    전체 개수로 나눈것. 즉 편차의 제곱의 평균을 말한다.

  - 표준편차(standard deviation)

    분산의 제곱근, 즉 분산 값에 루트를 씌운것을 말하며
    stdev라고 함.

최소 제곱법

- **공부한 시간의 차이에 따른 시험 성적 예측하기**
  - y절편 b값 구하기

    b = y의 평균 – (기울기 a x (x의 평균))

    b = 90.5 – (2.3 x 5) = 79

    최적의 직선(방정식)

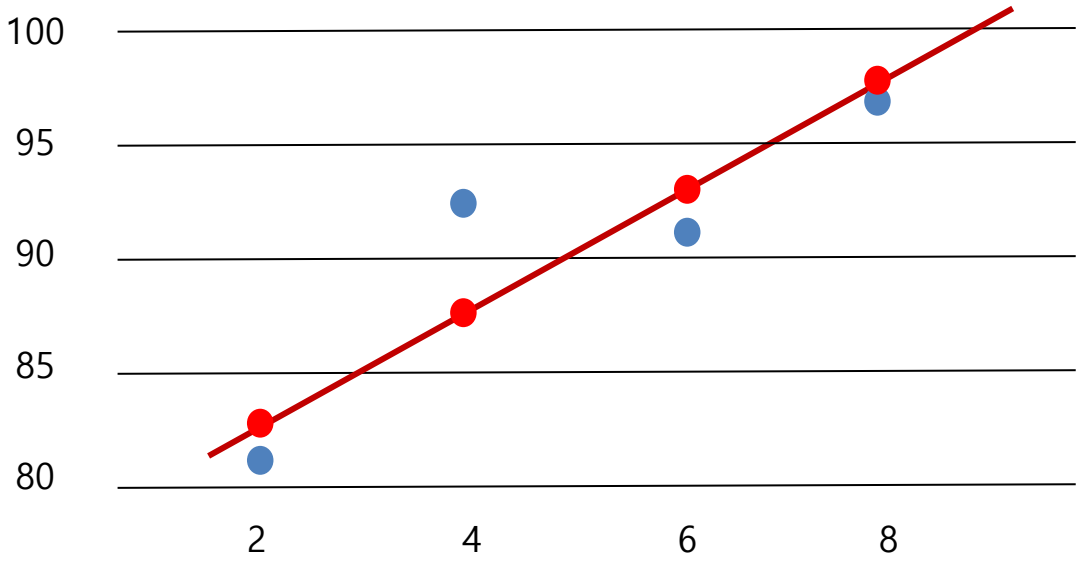
    $y = 2.3x + 79$

공부 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4

최소 제곱법

- 공부한 시간의 차이에 따른 시험 성적 예측하기

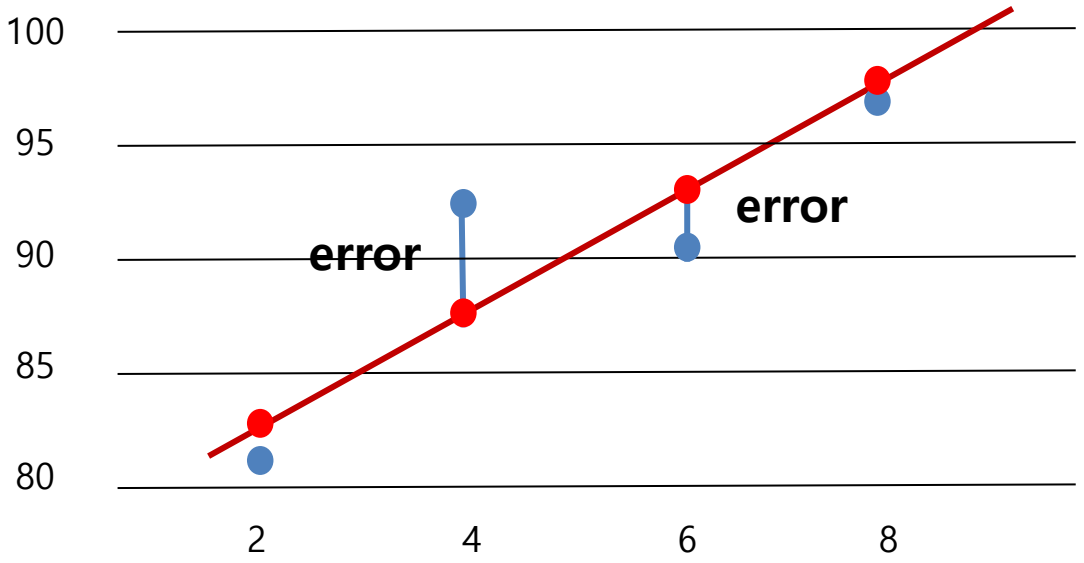
공부 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4



평균 제곱 오차

- **평균 제곱 오차**

선형 회귀는 임의의 직선을 그어 이에 대한 평균 제곱 오차를 구하고, 이 값을 가장 작게 만들어주는 기울기(a)와 절편(b)을 찾아가는 작업이다.



평균 제곱 오차

- **성능 평가 - 평균 제곱 오차 / 평균 절대값 오차**

  - MSE(Mean Squared Error)
    - 평균 제곱오차
    - 오차의 제곱을 평균으로 나눈 것

$$MSE = \frac{\sum_{i=1}^{n}(y-\hat{y})^2}{n}$$

  - MAE(Mean Absolute Error)
    - 평균 절대값 오차
    - 오차의 차이를 절대값으로 변환한 뒤 합산

$$MAE = \frac{\sum |y-\hat{y}|}{n}$$

평균 제곱 오차

- **오차 = 실제값 – 예측값**

공부 시간	2	4	6	8
성적	81	93	91	97
예측 값	82	88	94	100
오차	**1**	**-5**	**3**	**3**

분모 : 1 + 25 + 9 + 9 = 44

MSE = 44 / 4 = 11

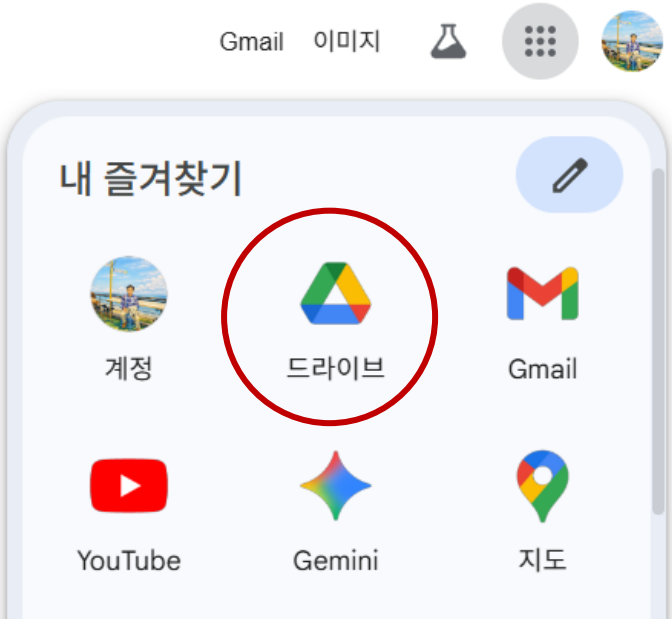
MAE = 1+5+3+3 = 12

구글 코랩

❖ Google Colab 사용하기

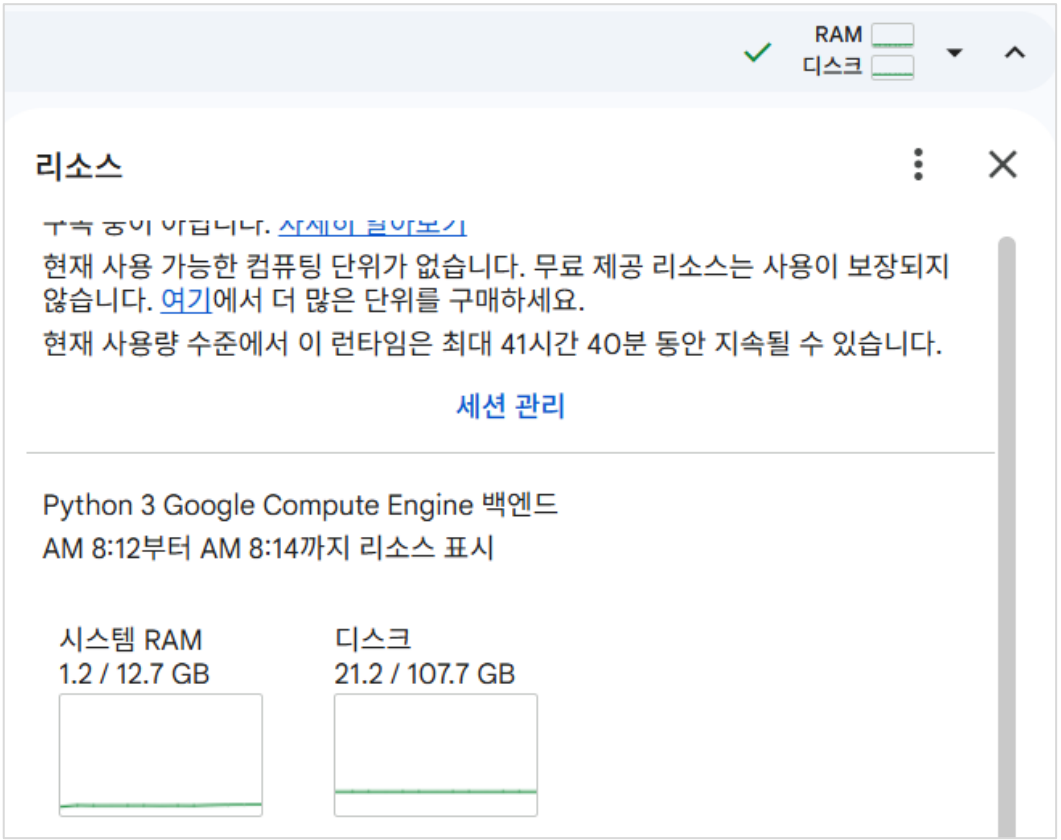
Colaboratory는 **구글의 Jupyter Notebook 기반 클라우드 서비스**로, TPU나 GPU, CPU를 무료 제공하며 클라우드 IDE의 일부이다

- 계정 로그인 > 메뉴(선택) > **드라이브**



구글 코랩

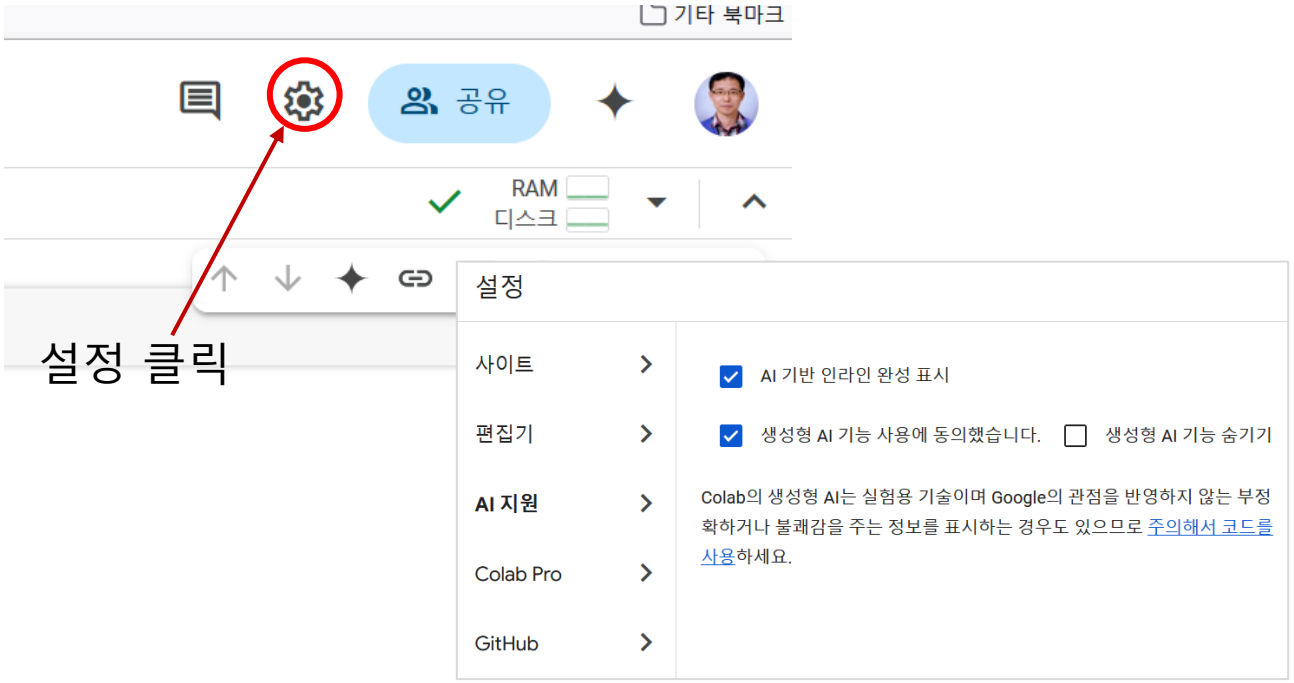
❖ 리소스 사용 – 클라우드 컴퓨터 사용



구글 코랩

❖ 구글 코랩(Colaboratory)

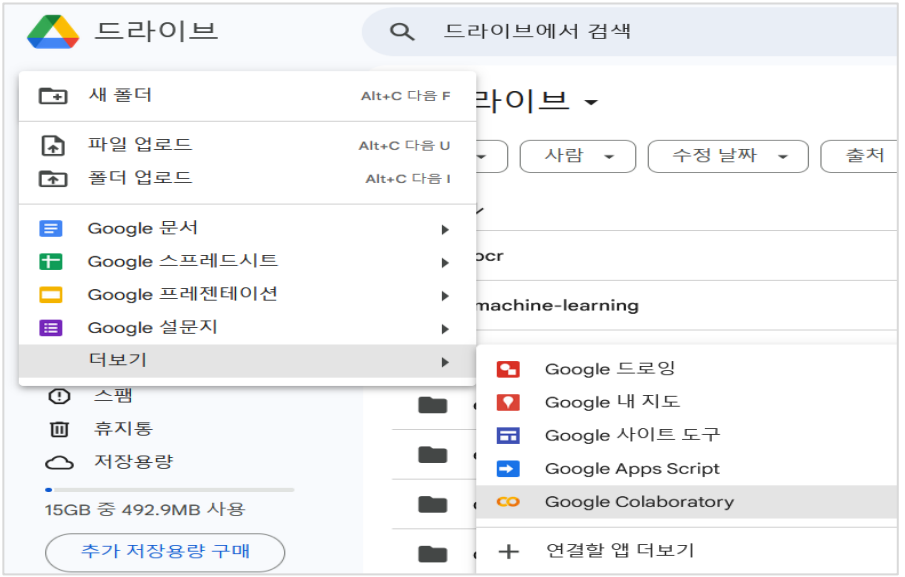
AI(Gemini) 지원 : 설정 > AI 지원 > 동의



구글 코랩

❖ 새 노트 만들기

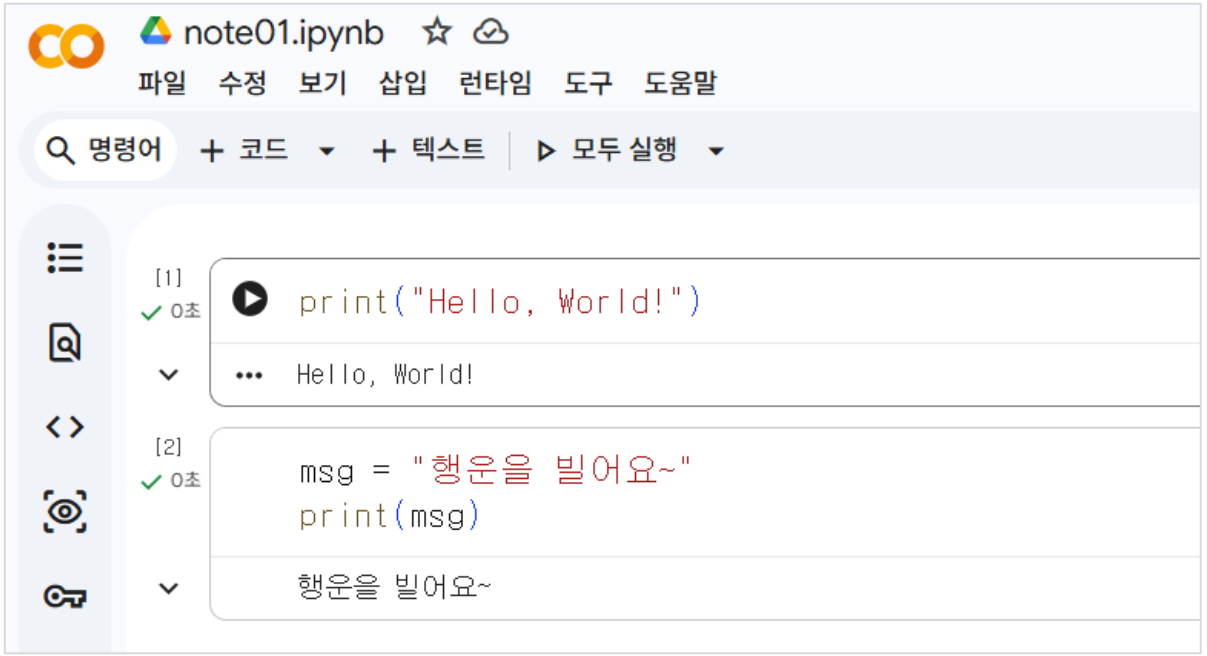
- 드라이브 > 홈
- + 신규 > 더보기 > Google Colaboratory



구글 코랩

❖ 새 노트 만들기

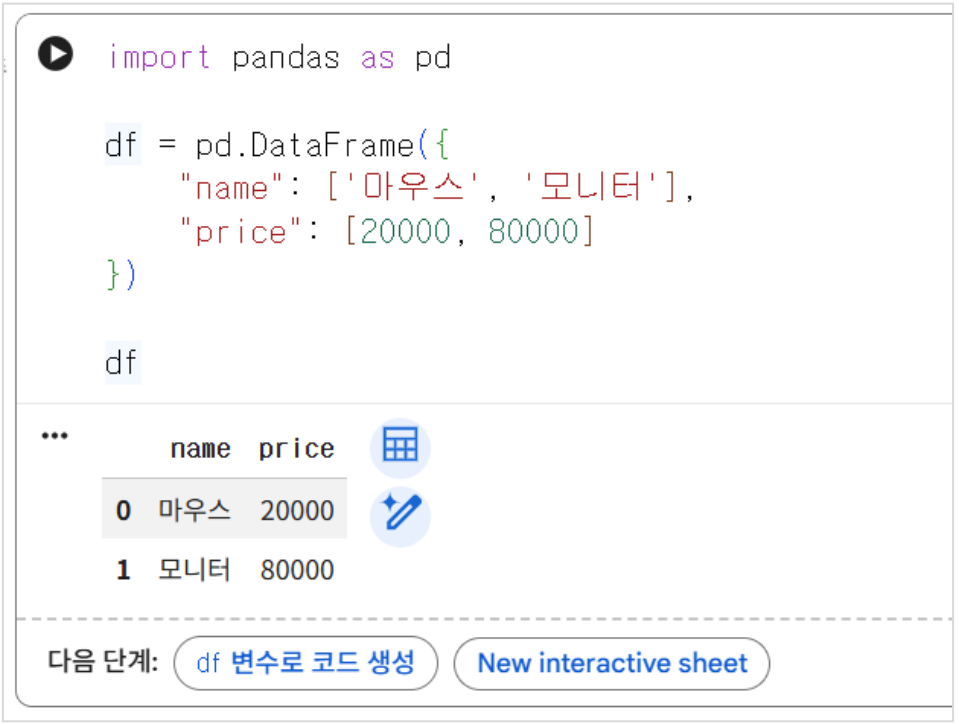
- 파일이름 > note01.ipynb



구글 코랩

❖ 새 노트 만들기

- 파일이름 > note01.ipynb



```
import pandas as pd

df = pd.DataFrame({
    "name": ['마우스', '모니터'],
    "price": [20000, 80000]
})

df
```

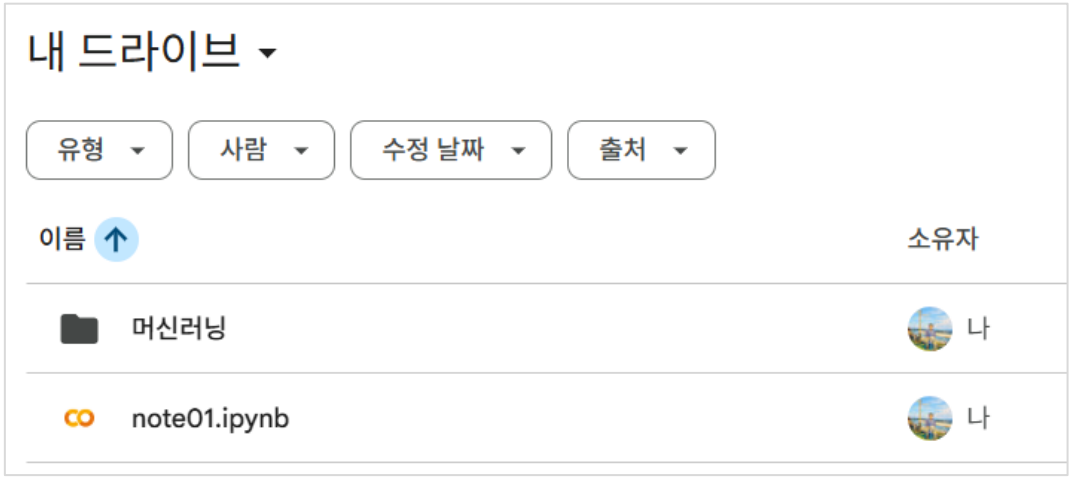
	name	price
0	마우스	20000
1	모니터	80000

다음 단계: df 변수로 코드 생성 | New interactive sheet

구글 코랩

❖ 새 폴더 만들기

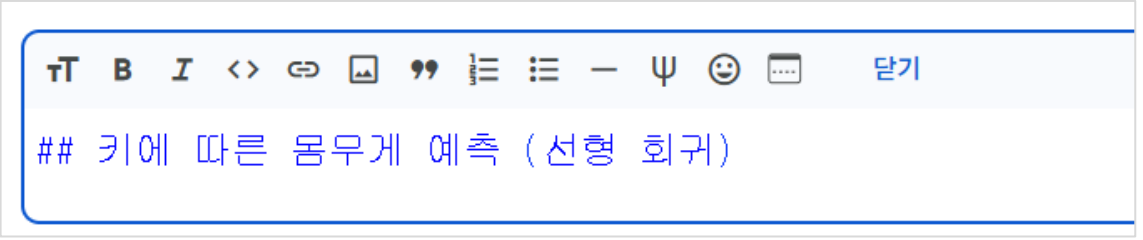
- 드라이브 > 내 드라이브
- + 신규 > 새폴더> 머신러닝



구글 코랩

❖ 텍스트 사용하기

- 파일 이름 : 선형회귀.ipynb
- 텍스트(버튼) > 문서의 제목을 만들때 사용



<편집화면>

shift + enter

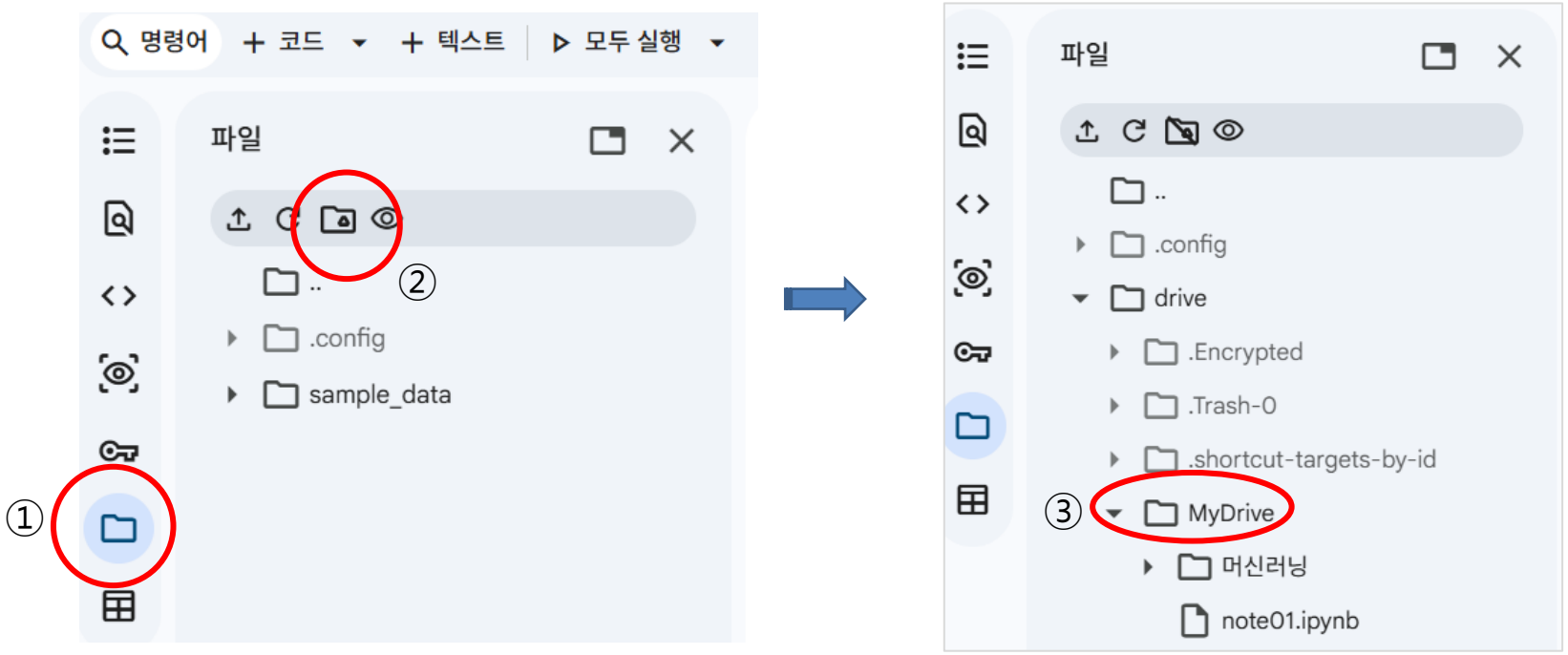
키에 따른 몸무게 예측 (선형 회귀)

<출력화면>

구글 코랩

❖ 드라이브 마운트

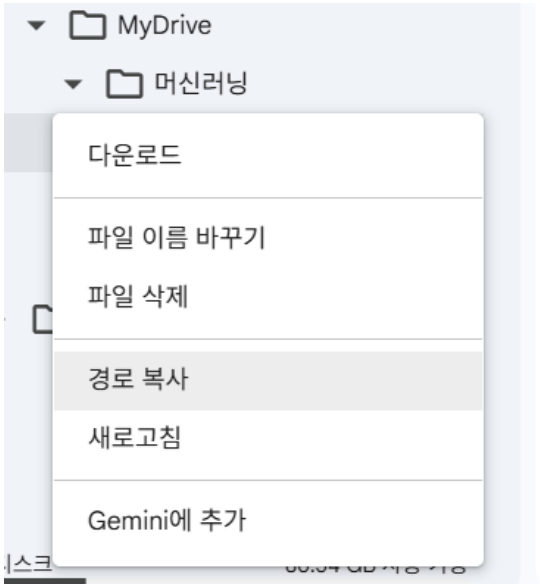
- 폴더 > 드라이브 마운트 > 드라이브 연결

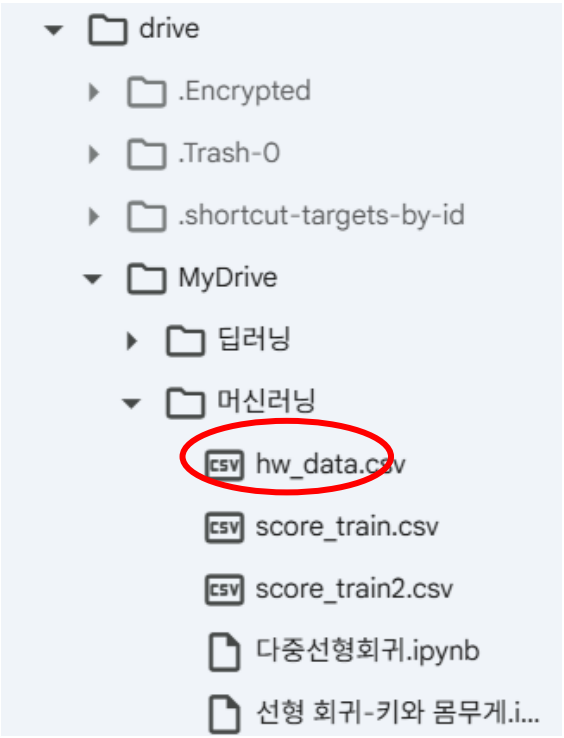


구글 코랩

❖ 훈련 데이터 업로드

- (...) 메뉴 > 업로드 > data_hw.csv





키에 따른 몸무게 예측

- 데이터 준비

```
import pandas as pd

# 가상의 키와 몸무게 데이터 생성
data_hw = {
    '키': [150, 160, 170, 175, 165, 155, 172, 168, 174, 158,
          162, 173, 185, 159, 167, 163, 171, 169, 180, 161],
    '몸무게': [42, 50, 70, 65, 60, 48, 68, 60, 65, 40,
              54, 67, 89, 51, 65, 60, 69, 71, 85, 53]
}
df_hw = pd.DataFrame(data_hw)

# csv 파일로 저장하기
df_hw.to_csv("/content/drive/MyDrive/Machine-Learning/hw_data.csv", index=False)

df_hw.head()
```

키에 따른 몸무게 예측

- 데이터 분할

```
# csv 파일 읽기
df_hw = pd.read_csv("/content/drive/MyDrive/Machine-Learning/hw_data.csv")
# print(df_hw)

# 변수 선택
X = df_hw[['키']] # 독립 변수 '키'(독립변수는 여러개일수 있으므로 이차원)
y = df_hw['몸무게'] # 종속 변수 '몸무게'

print("X (키):₩n", X.head())
print("₩ny (몸무게):₩n", y.head())

# 데이터 분할
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
          test_size=0.3, random_state=12) # 테스트 세트 크기를 20%로 변경

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

키에 따른 몸무게 예측

- **모델 학습 및 예측**

```
# 선형 회귀 모델 생성
from sklearn.linear_model import LinearRegression

model = LinearRegression()

# 모델 학습
model.fit(X_train, y_train)
print("키-몸무게 모델 학습이 완료되었습니다.")

# 모델 예측
pred = model.predict(X_train)
print("예측 결과 (일부):₩n", pred[:5])
```

```
키-몸무게 모델 학습이 완료되었습니다.
예측 결과 (일부):
 [55.96461345 71.56396569 46.8649913  59.86445151 53.36472141]
```

키에 따른 몸무게 예측

- 모델 시각화

```
# 훈련 데이터 시각화
import matplotlib.pyplot as plt

plt.scatter(X_train, y_train, label='Original Data')
plt.plot(X_train, pred, color='red', label='Regression Line')
plt.rc('font', family='NanumGothic') # 한글 글꼴 적용
plt.title('키에 따른 몸무게 선형 회귀')
plt.xlabel('키 (cm)')
plt.ylabel('몸무게 (kg)')
plt.legend()
plt.show()
```

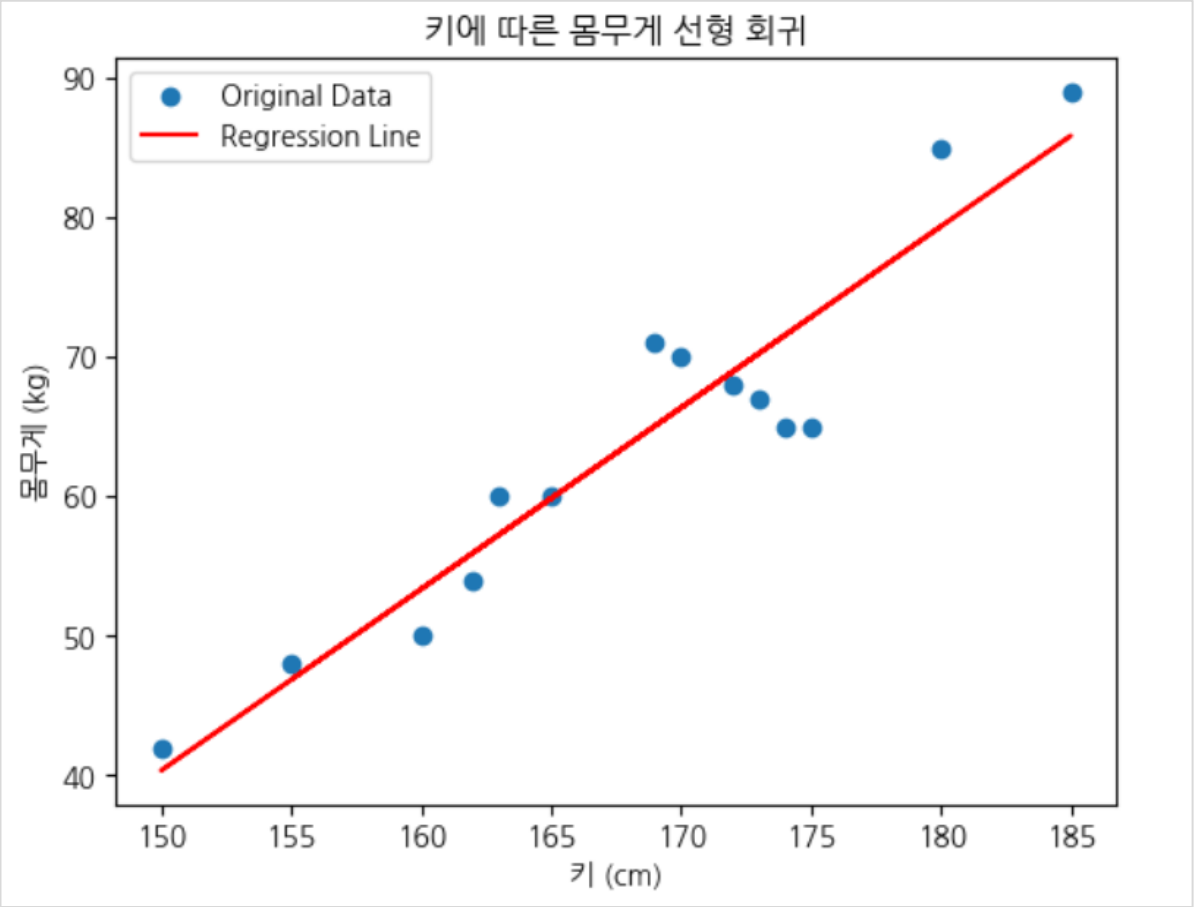
키에 따른 몸무게 예측

- **모델 시각화 – 한글 글꼴 설정**

```
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

# Install NanumGothic font
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv

# Add font to font manager
fm.fontManager.addfont('/usr/share/fonts/truetype/nanum/NanumGothic.ttf')

# Set the default font to NanumGothic
plt.rc('font', family='NanumGothic')

print("NanumGothic font installed and configured.")
```

키에 따른 몸무게 예측

- 모델 시각화



키에 따른 몸무게 예측

- 성능 평가 및 예측

```
# 직선의 방정식
print(f"기울기 (Coefficient): {model.coef_[0]:.4f}")
print(f"절편 (Intercept): {model.intercept_:.4f}")
# y = 1.299x - 154.6266

# 성능 평가 (전체 데이터)
# 결정 계수 - 1에 가까울수록 좋다.
r2_squared = model.score(X_test, y_test)
print(f"R2 Score: {r2_squared:.4f}")

# 몸무게 예측
height = 177
pred_weight = model.predict([[height]])
print(pred_weight)
```

```
R2 Score: 0.7440
[75.46380375]
```

키에 따른 몸무게 예측

- 성능 평가 및 예측

```
# 성능 평가 - mse, rmse (결과값이 작을 수록 좋다.)
from sklearn.metrics import mean_squared_error
import numpy as np

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred) #(실제값, 예측값)
print(f"평균 제곱 오차(MSE): {mse:.4f}")

rmse = np.sqrt(mse)
print(f"평균 제곱근 오차(RMSE): {rmse:.4f}")
```

```
평균 제곱 오차(MSE): 23.6951
평균 제곱근 오차(RMSE): 4.8678
```

딥러닝(Deep Learning)

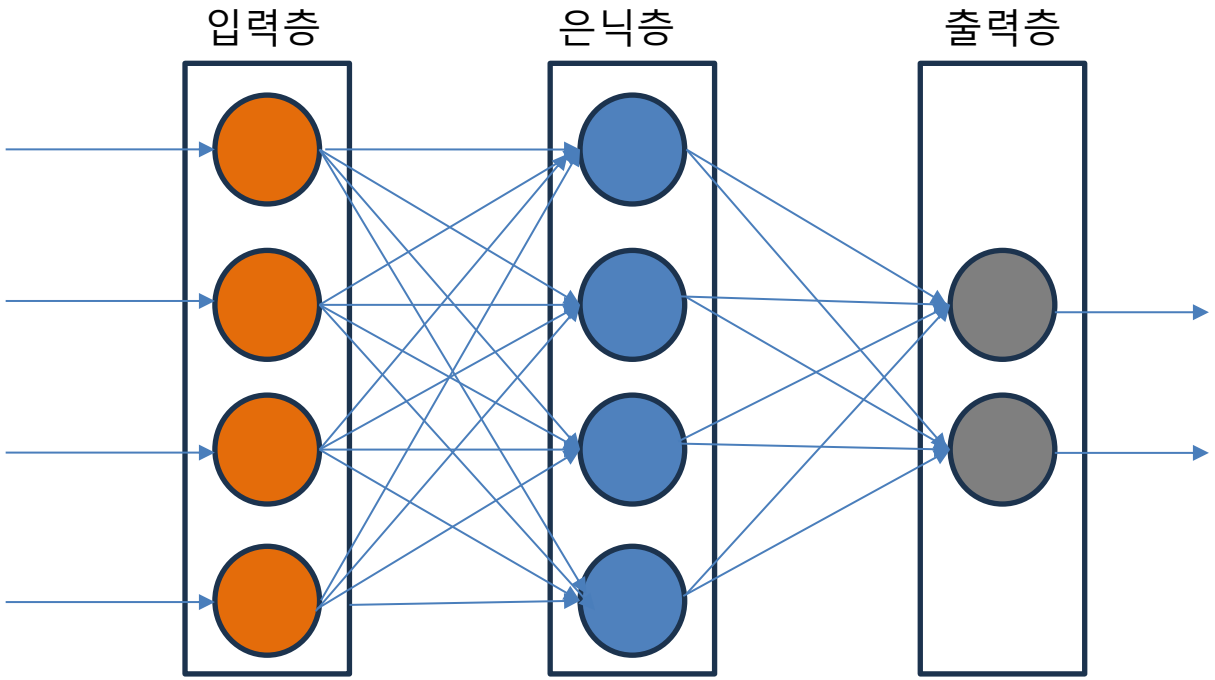
- **딥러닝이란?**

  딥러닝(Deep Learning)은 인공 신경망을 기반으로 하는 머신러닝의 한 분야입니다.
  사람의 뇌가 작동하는 방식과 유사하게, 여러 계층(layer)의 신경망을 통해 데이터를 학습하고 복잡한 패턴을 인식하여 예측이나 분류 등의 작업을 수행합니다.

  입력층 -> 은닉층 -> 은닉층 -> 출력층

딥러닝(Deep Learning)

- **인공 신경망**
  - **입력층(input layer)** – 데이터를 입력받는 층
  - **은닉층(hidden layer)** – 입력층에서 들어온 데이터가 여러 신호로 바뀌어 출력층까지 전달됨
  - **출력층(output layer)** – 출력층에 어떤 값이 전달되었는냐에 따라서 예측 값이 결정됨



딥러닝(Deep Learning)

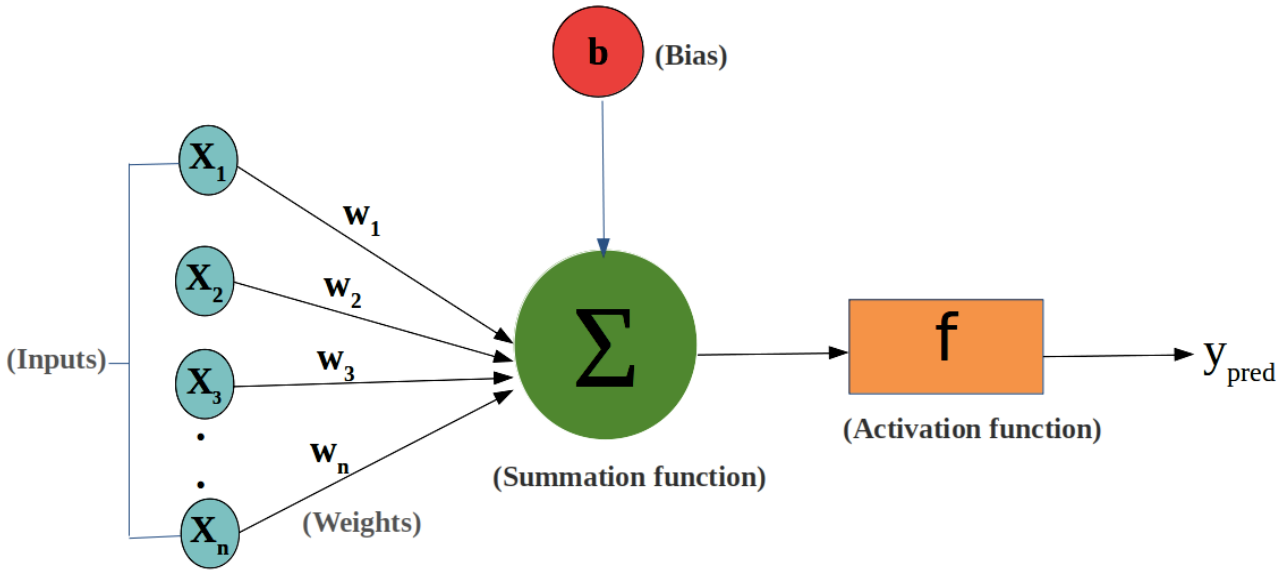
- **신호 전달 원리**

  인공신경망에서는 단순하게 신호를 전달해 주는 것이 아니라 신호 세기를 변경해서 전달합니다.
  신경망에서 중요한 용어 중 **가중치(weight)와 편향(bias)**이 있는데 이들이 바로 신호 세기를 변경하는 데 사용된다.
  한 뉴런에서 다음 뉴런으로 전달되는 신호 세기는 가중치와 편향에 의해 결정됨

  가중치란 그 값이 얼마나 중요한지 그렇지 않은지를 표현하기 위한 도구이고, **편향**은 한쪽으로 치우치는 값을 더할때 사용함

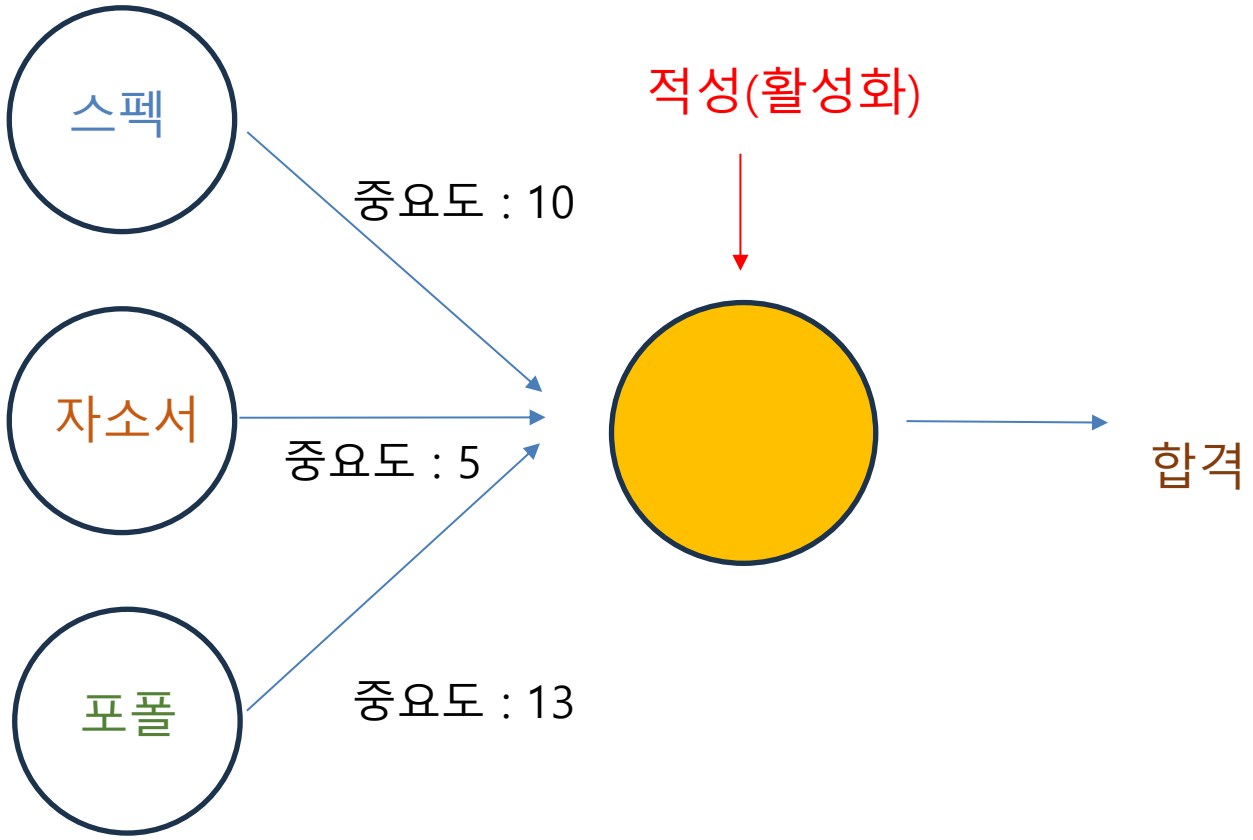
딥러닝(Deep Learning)

- 가중치와 편향



딥러닝(Deep Learning)

- 가중치와 편향

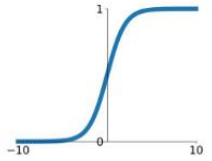


딥러닝(Deep Learning)

- **활성화 함수**
  - 뉴런을 활성화 할 필요가 있는지 없는 지 결정하는데 도움을 주는 함수
  - 비선형 문제를 해결하는데 중요한 역할을 한다

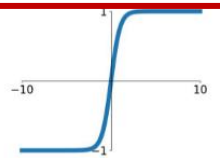
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



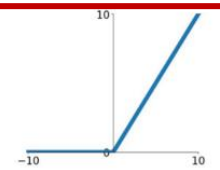
tanh

$$\tanh(x)$$



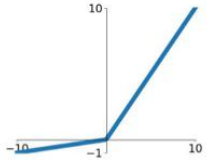
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

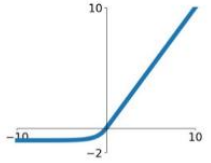


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



- **Softmax(소프트맥스) 함수**

  소프트 맥스 함수는 사실 활성화 함수는 아니고, 인공신경망의 마지막 부분 **출력층**에서 주로 사용되는 함수이다.

  최종 결과값을 정규화하는데 사용하는 함수가 소프트맥스 함수이다.
  이 소프트맥스 함수를 사용하면 예를 들어 남자와 여자를 구분할때 2개의 노드가 100% 확률. 60%, 40%라면 총합이 1(0.6, 0.4)이 되도록 보여준다.

머신러닝과 딥러닝 비교

⚙️ 머신러닝 vs 딥러닝 한눈에 비교

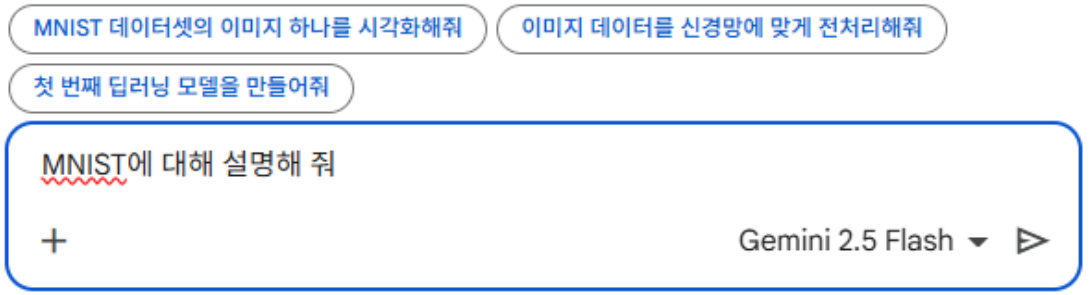
구분	머신러닝	딥러닝
데이터	적어도 가능	많이 필요
특징 추출	사람이 설계	자동
계산량	적음	매우 큼
성능	보통	매우 높음

🎯 한 줄 요약

- **머신러닝**: 데이터로 규칙을 배우는 AI
- **딥러닝**: 신경망을 깊게 쌓아 더 똑똑하게 만든 머신러닝

딥러닝(Deep Learning)

- **손글씨 예측 문제**

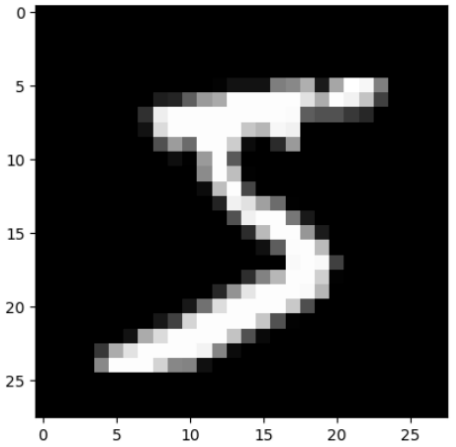


MNIST

미국 국립표준기술연구소에서 만든 손글씨 숫자 이미지를 혼합해서 만든 데이터 셋입니다.
머신러닝 및 딥러닝 분야에서 가장 널리 사용되는 데이터셋 중 하나입니다. 주로 손으로 쓴 숫자 이미지로 구성되어 있으며, 이미지 분류 모델의 성능을 평가하는 데 사용됩니다.

딥러닝(Deep Learning)

- **MNIST의 주요 특징**
  - **구성**: 0부터 9까지의 손글씨 숫자 이미지로 이루어져 있습니다.
  - **데이터 양**: 훈련 세트에 60,000개의 이미지, 테스트 세트에 10,000개의 이미 지가 포함되어 있습니다.
  - **이미지 크기**: 각 이미지는 28x28 픽셀의 흑백(회색조) 이미지입니다.



  - **용도**: 주로 이미지 분류, 특히 숫자 인식 분야에서 딥러닝 모델의 기본 성능 테스트 및 벤치마킹을 위해 활용됩니다. 간단하면서도 충분히 복잡하여 초보자들이 딥러닝을 학습하고 다양한 모델을 시도해보기 좋은 데이터셋입니다.

딥러닝(Deep Learning)

- **라이브러리 불러오기 및 데이터 준비**

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
import matplotlib.pyplot as plt
```

```
# 데이터 준비
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train.shape
```

```
(60000, 28, 28)
```

딥러닝(Deep Learning)

- **학습용 첫 데이터**

학습용 첫 데이터 보기

```
X_train[0]
```

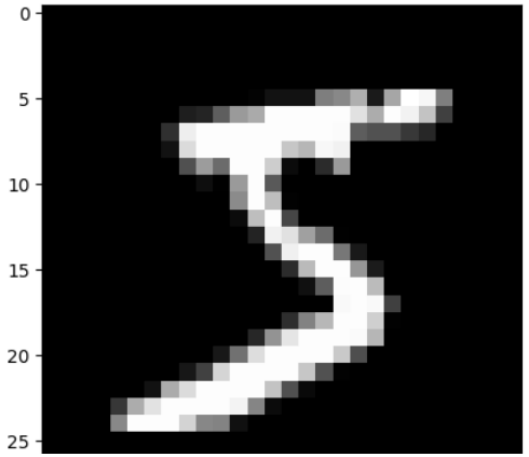
ndarray (28, 28) show data

```
print(y_train[0])
```

```
5
```

```
plt.imshow(X_train[0], cmap="gray") # 숫자를 이미지로 출력
```

```
<matplotlib.image.AxesImage at 0x7d006afc9160>
```



딥러닝(Deep Learning)

- **데이터 전처리**

mnist 데이터셋에서 데이터 X의 형태 바꾸기

(60000, 28, 28) -> (60000, 784)
인공 신경망의 입력층에 데이터를 넣을 때는 한 줄(행)로 만들어 넣어야 함
reshape() 사용

데이터 정규화

~~0255(색상) 사이의 값을 0~~1 사이의 값으로 바꾸기
RGB(255,255,255)- 흰색
RGB(0,0,0) - 검정색
RGB(255, 0, 0) - 빨간색

데이터를 255로 나눔 - 실수형으로 바뀜

```
X_train = X_train.reshape(60000, 784) #1차원 리스트 형태
X_test = X_test.reshape(10000, 784)

X_train = X_train / 255
X_test = X_test / 25
```

딥러닝(Deep Learning)

- **데이터 전처리**

```
X_train[0]
...
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.11764706, 0.14117647, 0.36862745, 0.60392157,
0.66666667, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
0.99215686, 0.88235294, 0.6745098 , 0.99215686, 0.94901961,
0.76470588, 0.25098039, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.19215686, 0.93333333,
0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
0.99215686, 0.99215686, 0.99215686, 0.98431373, 0.36470588,
0.32156863, 0.32156863, 0.21960784, 0.15294118, 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.07058824, 0.85882353, 0.99215686, 0.99215686,
0.99215686, 0.99215686, 0.99215686, 0.77647059, 0.71372549,
0.96862745, 0.94509804, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.31372549, 0.61176471, 0.41960784, 0.99215686, 0.99215686,
0.80392157, 0.04313725, 0.        , 0.16862745, 0.60392157,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.05490196
```

딥러닝(Deep Learning)

- **모델 만들기**

모델 만들기

입력층 - 은닉층(256) - 은닉층2(128) - 은닉층3(64) - 출력층(10)

```
# 모델 객체 생성
model = Sequential()

model.add(Dense(units=256, input_dim=784, activation="relu")) # 입력층
model.add(Dense(units=128, activation="relu")) # 은닉층1
model.add(Dense(units=64, activation="relu")) # 은닉층2
model.add(Dense(units=10, activation="softmax")) # 출력층
model.summary() # 요약
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	200,960
dense_1 (Dense)	(None, 128)	32,896
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 10)	650

```
Total params: 242,762 (948.29 KB)
Trainable params: 242,762 (948.29 KB)
Non-trainable params: 0 (0.00 B)
```

딥러닝(Deep Learning)

- **모델 학습**

```
# 모델 컴파일(최적화 함수, 손실함수, 평가 지표)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 모델 학습(훈련데이터, 정답, 반복횟수, 배치)
# batch_size : 100개의 샘플을 하나의 배치로 묶어서 모델에 입력
model.fit(X_train, y_train, epochs=5, batch_size=100)
```

```
Epoch 1/5
600/600 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.8591 - loss: 0.4954
Epoch 2/5
600/600 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 0.9665 - loss: 0.1073
Epoch 3/5
600/600 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.9797 - loss: 0.0677
Epoch 4/5
600/600 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - accuracy: 0.9849 - loss: 0.0493
Epoch 5/5
600/600 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 0.9889 - loss: 0.0362
<keras.src.callbacks.history.History at 0x79971a5ab2f0>
```

딥러닝(Deep Learning)

- **성능 평가 및 예측**

```
# 모델 평가(테스트 데이터, 정답)
loss, acc = model.evaluate(X_test, y_test)
print('Test Loss:', loss)
print('Test Accuracy:', acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - accuracy: 0.9701 - loss: 0.1122
Test Loss: 0.0929664894938469
Test Accuracy: 0.9739000201225281
```

```
# 숫자 예측
model.predict(X_test[0].reshape(1, 784))
# 9.9990523e-01 : 확률이 가장 높으므로 예측한 숫자는 7임
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
array([[8.2316537e-06, 4.5332445e-06, 1.0061069e-05, 3.4809689e-04,
        1.7134631e-08, 2.0233381e-06, 3.1317796e-11, 9.9909604e-01,
        5.4772568e-07, 5.3035503e-04]], dtype=float32)
```

딥러닝(Deep Learning)

- **숫자 이미지 출력**

```
# 예측한 숫자
pred_number = np.argmax(model.predict(X_test[0].reshape(1, 784)))
print(pred_number) #7

# 숫자 이미지 출력
plt.imshow(X_test[0].reshape(28, 28), cmap='gray') #출력시엔 이차원 형태로 변환
```