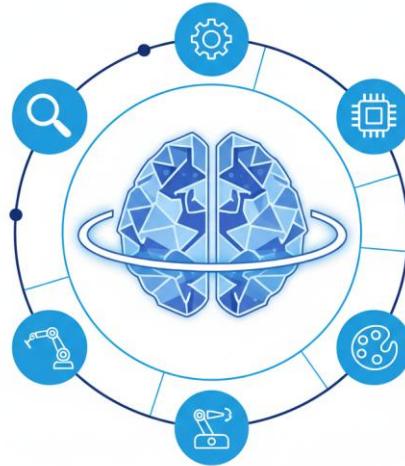


# 데이터 분석 및 머신러닝, 딥러닝

*chatGPT / Gemini / Copilot*



# 아나콘다(Anaconda) 플랫폼

## ❖ Anaconda Download

The screenshot shows the Anaconda website's download page. At the top, there is a navigation bar with the Anaconda logo and links for Products, Solutions, Resources, and Company. Below the navigation bar, the text "Choose Your" is displayed in large, bold, black letters. Underneath this, there are two tabs: "Windows" and "Mac", with a horizontal line separating them. A callout box highlights the "Anaconda Distribution" section, which contains the text: "Complete package with 8,000+ libraries, Jupyter, JupyterLab, and Spyder IDE. Everything you need for data science." Below this text is a blue button labeled "Windows 64-Bit Graphical Installer". This button is enclosed in a red rectangular box.

ANACONDA Products Solutions Resources Company

# Choose Your

Windows Mac

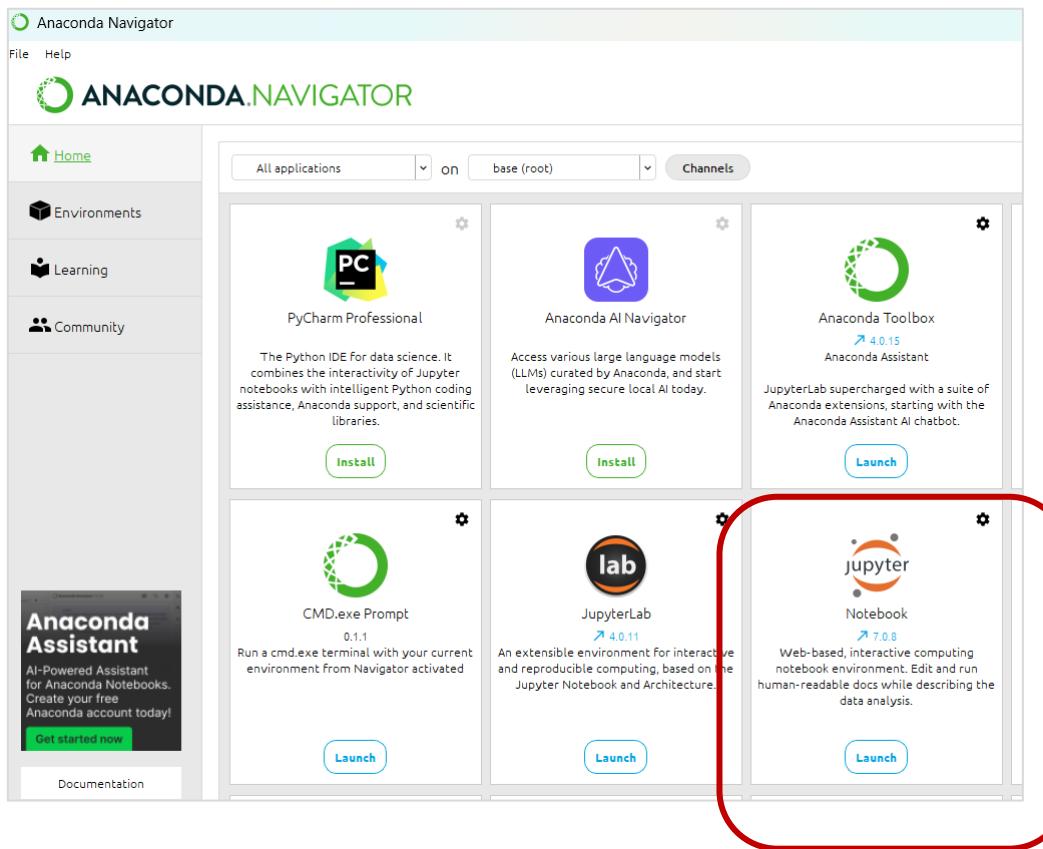
Anaconda Distribution

Complete package with 8,000+ libraries, Jupyter, JupyterLab, and Spyder IDE. Everything you need for data science.

Windows 64-Bit Graphical Installer

# 아나콘다(Anaconda) 플랫폼

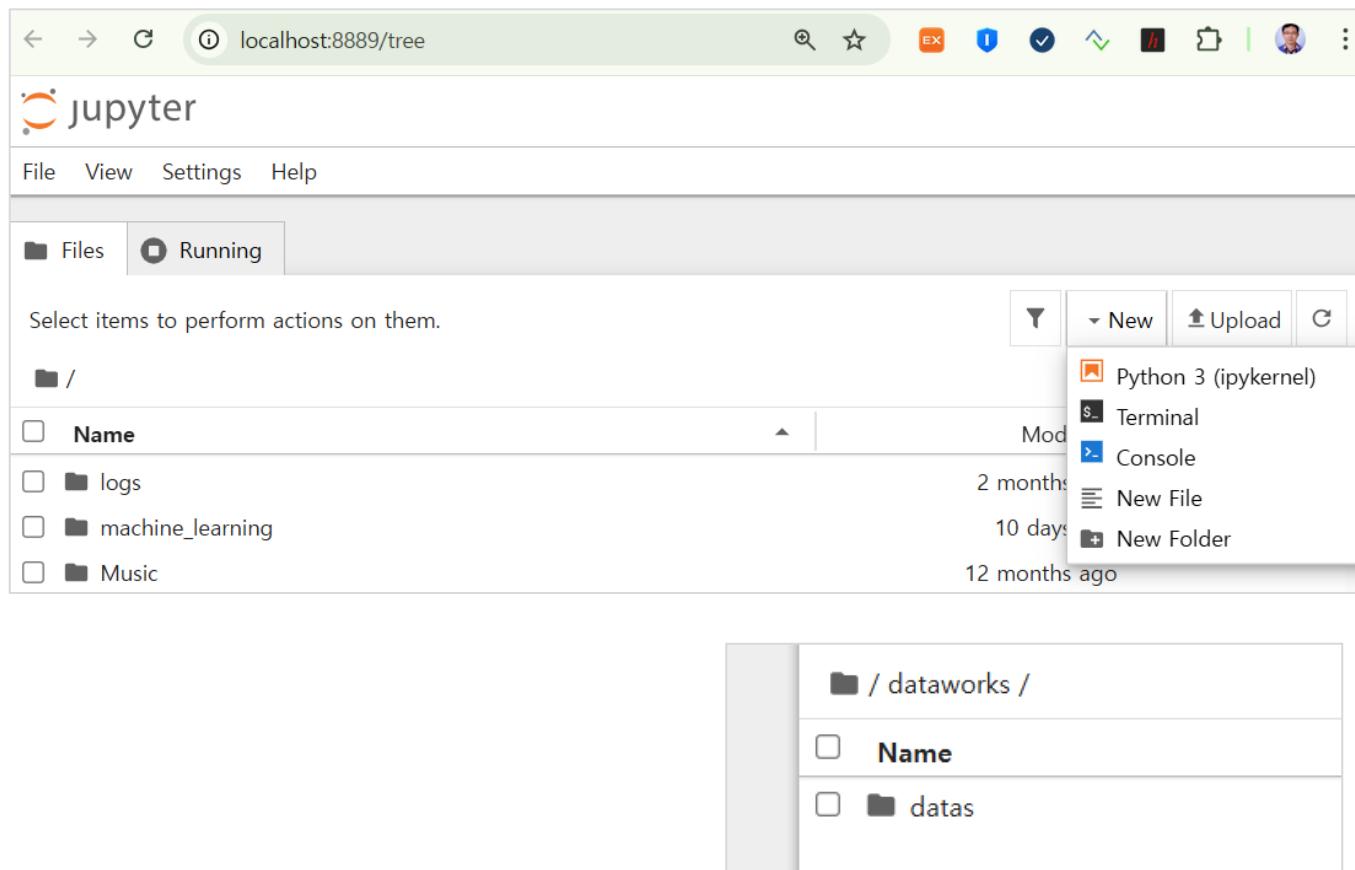
## ❖ 주피터 노트북(Jupyter Notebook) IDE 실행



# 주피터 노트북(Jupyter Notebook)

## ❖ 작업 디렉터리 생성

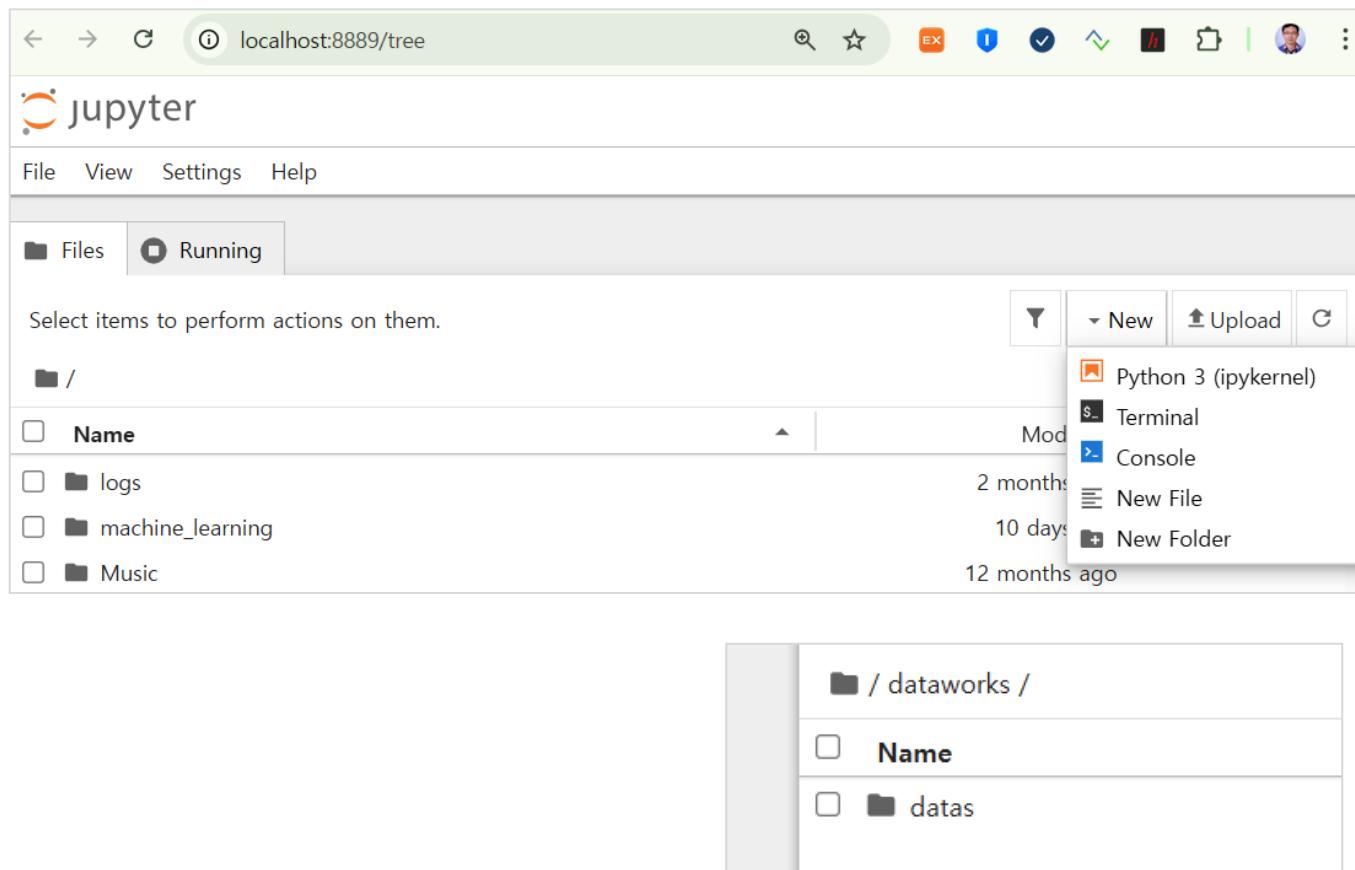
New > New Folder > dataworks 폴더 생성 > 하위에 datas 폴더 생성



# 주피터 노트북(Jupyter Notebook)

## ❖ 작업 디렉터리 생성

New > New Folder > dataworks 폴더 생성 > 하위에 datas 폴더 생성



# 주피터 노트북(Jupyter Notebook)

## ❖ 파일 생성(.ipynb)

The screenshot shows the Jupyter Notebook interface. At the top, there's a menu bar with File, View, Settings, and Help. Below the menu is a toolbar with 'Files' (selected), 'Running', 'Rename', and 'Delete'. A sidebar shows a directory structure with '/study\_data/' and subfolders 'Name' and 'datas'. On the right, a 'New' dropdown menu is open, listing options: New (disabled), Upload, Python 3 (ipykernel) (highlighted with a red box), Terminal, Console, New File, and New Folder. A red callout bubble points to 'Python 3 (ipykernel)' with the text '우측상단 New-> Python3'. Below the menu area, the main workspace shows a notebook cell with the title 'jupyter hello' and a status message 'Last Checkpoint: 2 minutes ago'. The cell content is '[1]: print("Hello~ World")' and the output is 'Hello~ World'. A blue callout bubble points to the cell content with the text '파일 이름' (File name). Another blue callout bubble points to the cell content with the text '코드 입력, 실행 : Run 또는 ctrl + Enter' (Code input, execution: Run or ctrl + Enter).

우측상단  
New->  
Python3

파일 이름

코드 입력,  
실행 : Run 또는  
ctrl + Enter

# 주피터 노트북(Jupyter Notebook)

## ❖ 코드 입, 출력

The screenshot displays a Jupyter Notebook interface with several code cells and their corresponding outputs:

- [1]: `print("Hello~ World")`  
Output: Hello~ World
- [2]: `print("안녕~ 세계")`  
Output: 안녕~ 세계
- [7]: `10 + 20`  
`10 - 20`  
Output: -10
- [6]:  
`n1 = 20`  
`n2 = 10`  
  
`print(n1 + n2)`  
`print(n1 - n2)`  
`print(n1 * n2)`  
`print(n1 / n2)`  
`print(n1 % n2)`  
Output:  
30  
10  
200  
2.0  
0
- [10]:  
`carts = ["계란", "라면", "콩나물"]`  
`print(carts)`  
  
`for cart in carts:`  
 `print(cart)`  
Output:  
['계란', '라면', '콩나물']  
계란  
라면  
콩나물

# 주피터 노트북(Jupyter Notebook)

## ❖ 세팅(Settings)

- 자동 완성 기능

**Home > Settings > Settings Editor > Enable autocompletion**

- 줄번호, 글꼴 등

**Home > Settings > Notebook**

- line numbers
- Font Size : 20



# 주피터 노트북(Jupyter Notebook)

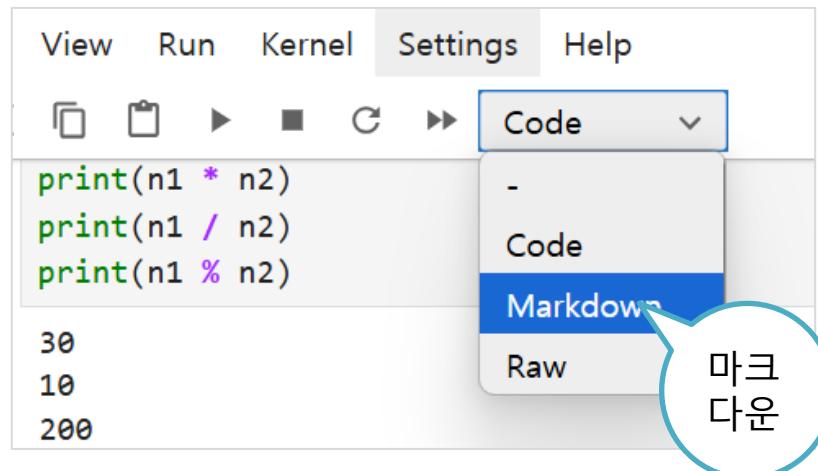
## ❖ Markdown

Markdown은 설명·정리·문서화를 담당하는 아주 중요한 기능이에요.

코드 옆에 글, 표, 수식, 이미지를 깔끔하게 적을 수 있어서 “노트 + 코드” 느낌으로 쓰기 딱 좋아요.

- 기호 종류

- 제목: #의 개수가 클수록 작은 제목( ## 텍스트)
- 목록 – '\*' 사용



마크  
다운

# 주피터 노트북(Jupyter Notebook)

## ❖ Markdown 편집 및 실행 화면

```
### 함수 만들기
* 기능 - 절대값 계산
* 설명 - 음수는 양수로 양수는 양수로 반환

def my_abs(x):
    if x < 0:
        return -x
    else:
        return x

print(my_abs(-3))
3
```

<편집 화면>

Ctrl + Enter

함수 만들기

- 기능 - 절대값 계산
- 설명 - 음수는 양수로 양수는 양수로 반환

```
def my_abs(x):
    if x < 0:
        return -x
    else:
        return x

print(my_abs(-3))
3
```

<실행 화면>

# 데이터 분석

## ● 데이터 분석용 주요 라이브러리

- + 데이터 분석을 위한 주요 라이브러리를 알려줘

0



라이브러리	주요 역할	핵심 기능
pandas	데이터 분석 핵심	DataFrame, CSV/Excel 처리, 필터링, 그룹화
numpy	수치계산 기반	다차원 배열, 수학/통계 연산, 고속 연산
matplotlib	데이터 시각화	선/막대/산점도 그래프



# 데이터 분석

## ● 판다스

- 데이터를 수집, 정리하는 데 최적화된 도구로 **표 형태(행·열)의 데이터**를 빠르고 편리하게 처리하기 위한 라이브러리입니다.
- 가장 인기있는 프로그래밍 언어인 **파이썬(python)**을 기반으로 한다.
- 시리즈(Series)와 데이터프레임(DataFrame)이라는 구조화된 데이터 형식을 제공한다.
- 판다스 설치 - **아나콘다 플랫폼**에는 이미 설치되어 있습니다.  
(`pip install pandas`)

# 데이터 분석

## ● 판다스 자료구조

- **시리즈(Series)** – 데이터가 순차적으로 나열된 1차원 배열의 형태를 가지며, 인덱스(Index)는 데이터 값(value)과 일대일 대응이 된다.
- **데이터프레임(DataFrame)** – 행(Row)과 열(Column)로 만들어지는 2차원 배열 구조로 표(Table) 형태이다.

	제품명
0	키보드
1	마우스
2	USB 메모리

인덱스  
(index)

[시리즈]

	제품명	가격
0	키보드	20,000
1	마우스	32,000
2	USB 메모리	10,000

칼럼  
(column)

[데이터프레임]

# 데이터 분석

- Series(시리즈)

```
import pandas as pd(별칭)
```

**pd.Series(데이터, 인덱스)**

속성	기능
index	- 시리즈의 인덱스 (생략하면 기본 0, 1, 2)
values	- 시리즈의 값(요소)
shape	- 시리즈의 크기 (요소의수,)

# 데이터 분석

## ● Series(시리즈)

### 시리즈(Series) 만들기

```
import pandas as pd

data = ['키보드', '마우스', 'USB 메모리']

s1 = pd.Series(data, index=[1, 2, 3])
print(s1)
print(type(s1))
```

```
# Series의 속성
print(s1.index)
print(s1.values)
print(s1.shape)
```

```
# 요소 선택 - 리스트 처럼 순서가 아닌 라벨 인덱스
print(s1[1])
print(s1[2])
```

```
1      키보드
2      마우스
3    USB 메모리
dtype: object
<class 'pandas.core.series.Series'>
Index([1, 2, 3], dtype='int64')
['키보드' '마우스' 'USB 메모리']
(3,)
키보드
마우스
```

# 데이터 분석

- DataFrame(데이터프레임)

```
import pandas as pd(별칭)
```

```
pd.DataFrame(데이터, 인덱스)
```

속성	기능
index	- 데이터프레임의 인덱스 (생략하면 기본 0, 1, 2)
columns	- 데이터프레임의 칼럼
shape	- 데이터프레임의 크기 (행, 열)

# 데이터 분석

칼럼(Column) 다루기

칼럼 추가 및 칼럼명 변경

```
: import pandas as pd

v df = pd.DataFrame({
    "name": ['키보드', '마우스', 'USB 메모리'],
    "price": [20000, 32000, 11000]
})

# 칼럼 추가
df['spec'] = ['유선', '무선', '128GB']

df['할인가'] = df['price'] * 0.8
print(df)
print("-----")

# 수정후 반드시 df에 다시 저장.
df = df.rename(columns={"name": "품명", "price": "가격", "spec": "제품상세"})
print(df)
```

	name	price	spec	할인가
0	키보드	20000	유선	16000.0
1	마우스	32000	무선	25600.0
2	USB 메모리	11000	128GB	8800.0

	품명	가격	제품상세	할인가
0	키보드	20000	유선	16000.0
1	마우스	32000	무선	25600.0
2	USB 메모리	11000	128GB	8800.0

# 데이터 분석

- 데이터 프레임을 CSV 파일로 변환

**CSV(Comma Separated Value) 파일**: 쉼표(,)로 데이터를 구분하여 저장하는 텍스트 파일 형식이다. 주로 표형태의 데이터를 저장하는 데 사용하며 엑셀에서 쉽고 열고 편집할 수 있음

☞ CSV 파일로 변환

```
df.to_csv("computer.csv", index=False)
```

☞ CSV 파일 읽기

```
pd.read_csv("computer.csv")
```

# 데이터 분석

## ● 데이터 프레임을 CSV 파일로 변환

CSV 파일이란?

쉼표(,)로 데이터를 구분하여 저장하는 텍스트 파일 형식이다.

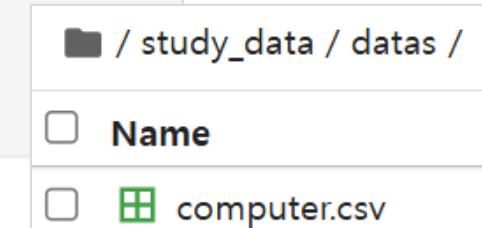
엑셀에서 .csv 형식으로 저장하고 불러올 수 있다.

```
import pandas as pd

# csv 파일 만들기 - index 제외
df.to_csv("datas/computer.csv", index=False)

# csv 파일 읽기
print(pd.read_csv("datas/computer.csv"))
```

	품명	가격	제품상세	할인가
0	키보드	20000	유선	16000.0
1	마우스	32000	무선	25600.0
2	USB 메모리	11000	128GB	8800.0



# 데이터 관리

## ● 행과 열 선택

속성	범위	예시
<code>loc[인덱스명, 칼럼명] (location)</code>	끝 인덱스 포함	인덱스 - 라벨인덱스
<code>iloc[인덱스번호, 칼럼번호] (integer location)</code>	끝 인덱스 - 1	인덱스 – 순서(0부터 시작)

# 데이터 관리

- 행/열 선택 : loc[행이름, 열이름] 사용

```
# csv 파일 읽어서 df 객체에 저장
df = pd.read_csv("datas/computer.csv")

# 행 선택 - Loc[행(행이름)]
print(df.loc[0:1])
print("=====")
```

```
# 특정 요소 선택 - Loc[행이름, 열(칼럼명)]
print(df.loc[0, "품명"]) #키보드
print(df.loc[1, "가격"]) #32000
print(df.loc[0:1, "품명"])
print(df.loc[0:1, ["품명", "가격"]])
```

```
품명      가격  제품상세      할인가
0 키보드  20000   유선  16000.0
1 마우스  32000   무선  25600.0
=====
키보드
32000
```

```
0    키보드
1    마우스
Name: 품명, dtype: object
품명      가격
0  키보드  20000
1  마우스  32000
```

# 데이터 관리

- 행/열 선택 : iloc[행번호, 열번호] 사용

```
# 행 선택 - iloc[행(인덱스번호)]
print(df.iloc[0:2]) # 끝인덱스-1
print("=====")

# 특정 요소 선택 - Loc[행(인덱스번호), 열(인덱스번호)]
print(df.iloc[0, 0]) #키보드
print(df.iloc[1, 1]) #32000
print(df.iloc[0:2, 0])
print(df.iloc[0:2, 0:2])
```

	품명	가격	제품상세	할인가
0	키보드	20000	유선	16000.0
1	마우스	32000	무선	25600.0

```
=====
키보드
32000
0    키보드
1    마우스
Name: 품명, dtype: object
품명      가격
0  키보드  20000
1  마우스  32000
```

# 데이터 관리

- 데이터 프레임 - 요소값 변경(업데이트)

`df.loc[행이름, 열이름] = "변경할 값"`

`df.iloc[행번호, 열번호] = "변경할 값"`

- 데이터 프레임 - 요소값 추가

`df[열] = "추가할 칼럼"` => 열 추가

`df.loc[행] = "추가할 값"` => 행 추가

# 데이터 관리

## ● 데이터 프레임 - 행/열 요소 변경 및 추가

```
# 요소값 변경 및 추가
df2 = df.copy() #원본 유지 복사
df2

# 품명 '마우스'를 '무선마우스'로 변경
df2.loc[1, "품명"] = "무선마우스"
# 무선 마우스의 가격을 35000으로 변경
df2.loc[1, "가격"] = 35000
# 무선 마우스의 할인가 수정
df2.loc[1, "할인가"] = df2.loc[1, "가격"] * 0.8
df2

# 행 추가
df2.loc[3] = ["키보드", 40000, "무선", 0.0]
df2.loc[3, "할인가"] = df2.loc[3, "가격"] * 0.8
df2
```

	품명	가격	제품상세	할인가
0	키보드	20000	유선	16000.0
1	무선마우스	35000	무선	28000.0
2	USB 메모리	11000	128GB	8800.0
3	키보드	40000	무선	32000.0

# 데이터 관리

- 데이터 프레임 - 행/열 삭제

**df.drop(행, axis=0) => 행 삭제**

**df.drop(열, axis=1) => 열 삭제**

```
# 행 삭제 - drop(행번호, axis=0)
# 열 삭제 - drop(칼럼명, axis=1)
df3 = df2.copy()
df3

# 3행 삭제
df3 = df3.drop(3, axis=0)
df3

# 열 삭제
# df3 = df3.drop("할인가", axis=1)
df3 = df3.drop(["제품상세", "할인가"], axis=1)
df3
```

	품명	가격
0	키보드	20000
1	무선마우스	35000
2	USB 메모리	11000

# 데이터 탐색

## ● 데이터 프레임을 CSV 파일로 변환

```
# 데이터프레임 만들기
food = pd.DataFrame({
    "메뉴" : ["김밥", "비빔밥", "자장면", "우동", "김밥", "자장면", "김밥"],
    "가격" : [3000, 7000, 5500, 5500, 3500, 6000, 4000],
    "분류" : ["한식", "한식", "중식", "일식", "한식", "중식", "한식"]
})
# print(food)

# csv 파일로 변환
food.to_csv("food.csv", index=False)

# csv 파일 읽어오기
df = pd.read_csv("food.csv")
print(df)
```

# 데이터 탐색

## ● 데이터 탐색을 위한 주요 함수

```
# 데이터 출력 - head(), tail() -> 기본 5개
print(df.head())
print(df.tail())

# 데이터 정보 - info()
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  -- 
 0   메뉴      7 non-null    object 
 1   가격      7 non-null    int64  
 2   분류      7 non-null    object 
dtypes: int64(1), object(2)
memory usage: 300.0+ bytes
```

# 데이터 탐색

## ● 데이터 탐색을 위한 주요 함수

```
# 고유한 값의 개수 - value_counts()  
print(df["메뉴"].value_counts())  
print(df["분류"].value_counts())  
  
# 통계 요약 - describe()  
# print(df.describe())  
print(df.describe(include='number'))  
print(df.describe(include='object'))
```

메뉴	
김밥	3
자장면	2
비빔밥	1
우동	1

Name: count, dtype: int64

분류	
한식	4
중식	2
일식	1

Name: count, dtype: int64

가격	
count	7.000000
mean	4928.571429
std	1455.694892
min	3000.000000
25%	3750.000000
50%	5500.000000
75%	5750.000000
max	7000.000000

# 데이터 탐색

## ● 조건 검색(필터링)

비교 연산 기호	설명
>, >=	크다, 크거나 같다.
<, <=	작다, 작거나 같다
==	두 항이 같다.
!=	두 항이 같지 않다.

논리 연산	연산 기호	설명
교집합(AND)	&	두 조건이 모두 참일때 True 반환
합집합(OR)		두 조건중 하나 이상이 참일때 True 반환
부정(NOT)	~	조건이 참이면 False, 거짓이면 True를 반환

# 데이터 탐색

## ● 데이터 탐색 – 조건 검색(필터링)

```
# 조건 검색(필터링)
# 메뉴가 '자장면'인 음식의 정보
result1 = (df["메뉴"] == '자장면')
print(df[result1])

# 메뉴가 '김밥'이 아닌 음식의 정보
result2 = (df["메뉴"] != '김밥')
# result2 = ~(df["메뉴"] == '김밥')
print(df[result2])

# 가격이 4000원 미만인 음식의 정보
result3 = (df["가격"] < 4000)
print(df[result3])

# 메뉴가 '자장면'이고, 가격이 6000원 이상인 음식의 정보
# result4 = (df["메뉴"] == '자장면') & (df["가격"] >= 6000)
result4 = (df["메뉴"] == '자장면') | (df["가격"] >= 6000)
print(df[result4])
```

# 데이터 탐색

## ● 데이터 탐색 - 정렬

정렬 분류	함수	파라미터
칼럼 기준	sort_values(칼럼, 파라미터)	오름차순: ascending=True
인덱스 기준	sort_index(파라미터)	내림차순: ascending=False

	메뉴	가격	분류
1	비빔밥	7000	한식
5	자장면	6000	중식
2	자장면	5500	중식
3	우동	5500	일식
6	김밥	4000	한식
4	김밥	3500	한식
0	김밥	3000	한식

	메뉴	가격	분류
6	김밥	4000	한식
4	김밥	3500	한식
0	김밥	3000	한식
1	비빔밥	7000	한식
3	우동	5500	일식
5	자장면	6000	중식
2	자장면	5500	중식

# 데이터 탐색

## ● 데이터 탐색 - 정렬

```
df = pd.read_csv("datas/food.csv")
# print(df)
# print("=====")

# 가격을 내림차순으로 정렬하기
df.sort_values("가격", ascending=False, inplace=True)
print(df)
print("=====")

# 메뉴를 오름차순으로 정렬, 메뉴가 같으면 가격을 내림차순 정렬
df.sort_values(["메뉴", "가격"], ascending=[True, False], inplace=True)
print(df)
print("=====")

# 인덱스 기준 정렬(초기화)
df.sort_index(ascending=True)
print(df)
```

# 데이터 탐색

- 판다스의 통계, 수학 , 그룹핑 함수

함수	기능
count() / sum() / mean()	개수 / 합계 / 평균
var() / std()	분산 / 표준편차
max() / min()	최대값 / 최소값
idxmax() / idxmin()	최대값 위치 / 최소값 위치
quantile(.25) / quantile(.75)	1사분위수 / 3사분위수
mode()	빈도값
round(수, 자리수)	반올림
groupby(칼럼명)	그룹화

# 데이터 탐색

- 판다스의 통계, 수학 함수

```
print("개수:", df["가격"].count())
print("합계:", df["가격"].sum())
print("평균:", round(df["가격"].mean(), 2))
print("분산:", round(df["가격"].var(), 2))
print("표준편차:", round(df["가격"].std(), 2))
print("최대값:", df["가격"].max())
print("최소값:", df["가격"].min())
print("최대값의 위치:", df["가격"].idxmax())
print("최소값의 위치:", df["가격"].idxmin())
```

# 데이터 탐색

- 판다스의 통계, 수학 함수

```
# 사분위수 - quantile()
print("1사분위수:", df["가격"].quantile(.25))
print("2사분위수:", df["가격"].quantile(.50))
print("3사분위수:", df["가격"].quantile(.75))
print("4사분위수:", df["가격"].quantile(1.0))

# 조건 - 1사분위수 보다 작은 가격을 출력
result = df["가격"] < df["가격"].quantile(.25)
print(df[result])

# 최빈값 - mode()
print("최빈값:", df["가격"].mode()[0])
```

# 데이터 탐색

## ● 그룹핑 - groupby()

```
import pandas as pd

# "수량" 칼럼 추가
df["수량"] = [10, 10, 15, 5, 7, 7, 8]
df

# csv 파일로 변환
df.to_csv("./datas/food2.csv", index=False)

# csv 파일 읽기
food = pd.read_csv("./datas/food2.csv")
food
```

메뉴	가격	분류	수량
김밥	3000	한식	10
비빔밥	7000	한식	10
자장면	5500	중식	15
우동	5500	일식	5
김밥	3500	한식	7
자장면	6000	중식	7
김밥	4000	한식	8

# 데이터 탐색

- 그룹핑 - groupby()

```
food = pd.read_csv("food2.csv")
# print(food)

# 그룹화 - groupby()
df["분류"].value_counts()

food.groupby("메뉴").mean(numeric_only=True)
food.groupby("분류").mean(numeric_only=True) #모든 수치 칼럼 평균

food.groupby("분류")["가격"].mean() #분류별 가격의 평균
food.groupby("분류")["수량"].mean() #분류별 수량의 평균
```

# 데이터 전처리

## ● 결측치 삭제 및 대체

함수	설명
drop(행, axis=0)	NaN이 포함된 행 삭제
drop(열, axis=1)	NaN이 포함된 열 삭제
dropna()	NaN이 포함된 모든 행 삭제
fillna()	<ul style="list-style-type: none"><li>- 수치형: 평균(mean()), 중간값(median())</li><li>- 문자형: 빈도값(mode())</li></ul>
isnull()	<ul style="list-style-type: none"><li>- 결측치 확인(True / False로 반환)</li></ul>
isnull().sum()	<ul style="list-style-type: none"><li>- 결측치 개수(True(1))</li></ul>

# 데이터 전처리

## ● 결측치 대체

```
data = {  
    "메뉴": ["김밥", "비빔밥", "자장면", "우동", None, "자장면", "김밥"],  
    "가격": [3000, 7000, 5500, None, 3500, 6000, 4000],  
    "분류": ["한식", "한식", "중식", "일식", "한식", "중식", "한식"]  
}  
  
food = pd.DataFrame(data)  
food  
food.info()  
  
# 결측치 확인 - isnull()  
print(food.isnull())  
print(food.isnull().sum()) #True(1) / False(0)
```

메뉴	1
가격	1
분류	0

# 데이터 전처리

## ● 결측치 대체

```
# 수치형 - 중간값으로 대체
median = food["가격"].median()
median

# fillna()
food["가격"] = food["가격"].fillna(median)
food

# 문자형 - 최빈값
frequency = food["메뉴"].mode()[0]
frequency
food["메뉴"] = food["메뉴"].fillna(frequency)
food

print(food.isnull().sum())
```

# 데이터 시각화

## ● matplotlib 라이브러리

- + matplotlib 시각화 라이브러리에 대해 설명해줘



- 데이터 분석, 과학 시각화, 보고서 작성 등 다양한 분야에서 그래프를 그릴 때 사용됩니다.
- 주요 그래프 종류

종류	함수명	예시
선 그래프	plot()	plt.plot(x, y)
막대 그래프(수직/수직)	bar() / barh()	plt.bar(x, y) / plt.barh(x, y)
산점도	scatter()	plt.scatter(x, y)
히스토그램	hist()	plt.hist(data)
파이차트	pie()	plt.pie(data)

# 데이터 시각화

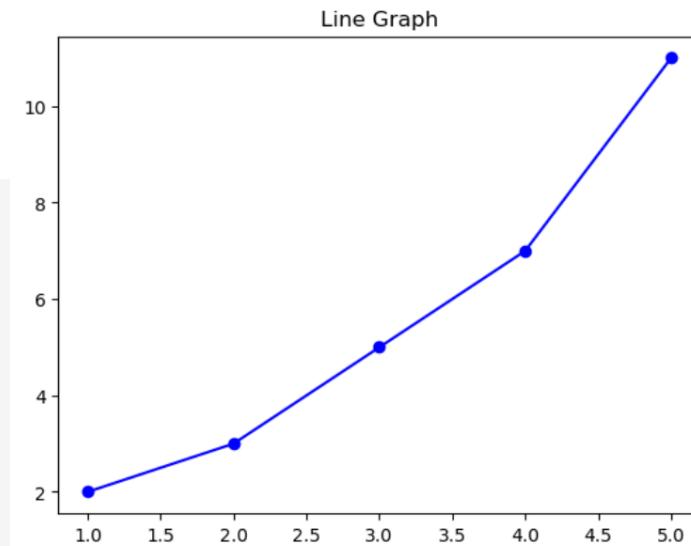
## ● 선 그래프

```
import matplotlib.pyplot as plt

# 데이터 준비
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# 선 그래프 그리기
plt.plot(x, y, label='Prime numbers', color='blue', marker='o')
plt.title('Line Graph') # 제목
plt.legend() # 범례

# 그래프 출력
plt.show()
```



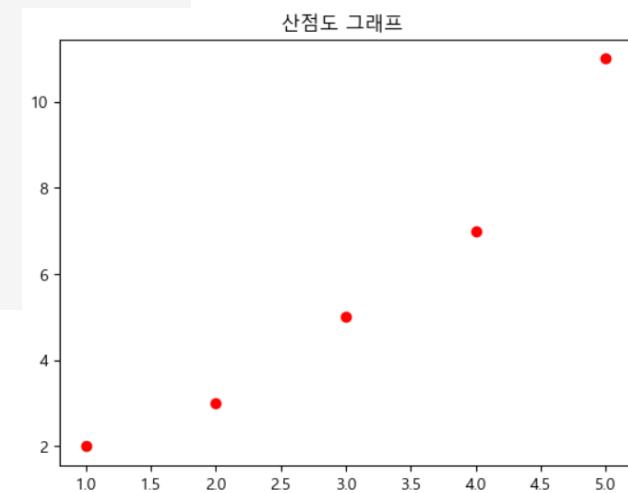
# 데이터 시각화

- 산점도 그래프

```
import matplotlib.pyplot as plt

# 데이터 준비
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11] # 소수

plt.rc('font', family='Malgun Gothic')
plt.title('산점도 그래프 ')
plt.scatter(x, y, color='red')
plt.show()
```



# 데이터 시각화

## ● 막대 그래프

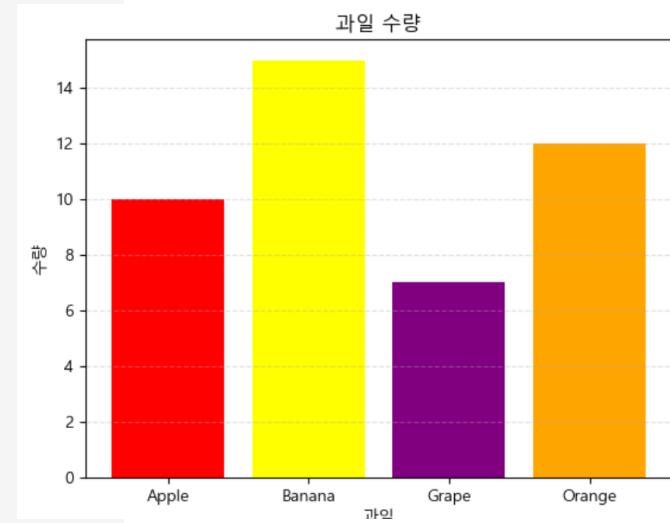
```
import matplotlib.pyplot as plt

# 데이터 준비
fruits = ['Apple', 'Banana', 'Grape', 'Orange']
counts = [10, 15, 7, 12]
color = ['red', 'yellow', 'purple', 'orange']

# 한글 깨짐 방지 - Batang, Gulim 사용
plt.rc('font', family="Malgun Gothic")

# 막대 그래프
plt.bar(fruits, counts, color=color)
plt.title('과일 수량')
plt.xlabel('과일') # x축 제목
plt.ylabel('수량') # y축 제목

# y축 grid 추가
# plt.grid()
plt.grid(axis='y', linestyle='--', alpha=0.3)
plt.show()
```



# 데이터 시각화

- **seaborn 라이브러리**

- Python의 데이터 시각화 라이브러리로, 통계적 그래프를 쉽게 그릴 수 있도록 matplotlib을 기반으로 만들어졌다
- pandas의 DataFrame과 잘 통합되어 있어, 데이터를 시각적으로 분석하고자 할 때 매우 유용하다

- 설치 방법

- pip install seaborn**



# 타이타닉호 사례 분석

## ● 타이타닉호 데이터 준비

```
import seaborn as sns  
  
df = sns.load_dataset('titanic')  
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no

# 타이타닉호 사례 분석

## ● 타이타닉호 데이터 정보(결측치 확인)

#	Column	Non-Null Count	Dtype
0	survived	891 non-null	int64
1	pclass	891 non-null	int64
2	sex	891 non-null	object
3	age	714 non-null	float64
4	sibsp	891 non-null	int64
5	parch	891 non-null	int64
6	fare	891 non-null	float64
7	embarked	889 non-null	object
8	class	891 non-null	category
9	who	891 non-null	object
10	adult_male	891 non-null	bool
11	deck	203 non-null	category
12	embark_town	891 non-null	object
13	alive	891 non-null	object
14	alone	891 non-null	bool



#	Column	Non-Null Count	Dtype
0	survived	891 non-null	int64
1	pclass	891 non-null	int64
2	sex	891 non-null	object
3	age	714 non-null	float64
4	sibsp	891 non-null	int64
5	parch	891 non-null	int64
6	fare	891 non-null	float64
7	embarked	889 non-null	object
8	class	891 non-null	category
9	who	891 non-null	object
10	adult_male	891 non-null	bool
11	embark_town	891 non-null	object
12	alive	891 non-null	object
13	alone	891 non-null	bool



# 타이타닉호 사례 분석

## ● 타이타닉호 데이터 분석 및 처리

```
# 결측치 처리  
# df.isnull().sum()  
print(df.isna().sum())  
  
# age, embarked, embark_town 칼럼 - 결측치 대체  
# deck 칼럼 - 결측치 삭제  
df2 = df.copy() #원본 복사  
  
# age - 평균값 계산  
mean_age = df2['age'].mean()  
print(mean_age) #29.699  
  
# 평균값으로 채우기  
df2['age'] = df2['age'].fillna(mean_age)  
  
# embarked - 최빈값 계산  
most_freq = df2['embarked'].mode()[0]  
print(most_freq) #S  
  
df2['embarked'] = df2['embarked'].fillna(most_freq)  
print(df2.isna().sum())
```

survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	0
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0
dtype:	int64

# 타이타닉호 사례 분석

## ● 타이타닉호 데이터 분석 및 처리

```
# embark_town - 최빈값 이용
print(df2['embark_town'][825:830])

most_freq = df2['embark_town'].mode()[0]
print(most_freq) #Southampton

# 최빈값으로 채우기
df['embark_town'] = df['embark_town'].fillna(most_freq)
print(df['embark_town'][825:830])

# deck 열 삭제
df2 = df2.drop('deck', axis=1)
df2.info()
print(df2.isna().sum())
df2.columns # 칼럼 확인
```

survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	0
class	0
who	0
adult_male	0
embark_town	0
alive	0
alone	0
dtype:	int64

# 타이타닉호 사례 분석

- 타이타닉호 - 성별에 따른 생존자 수 그래프

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

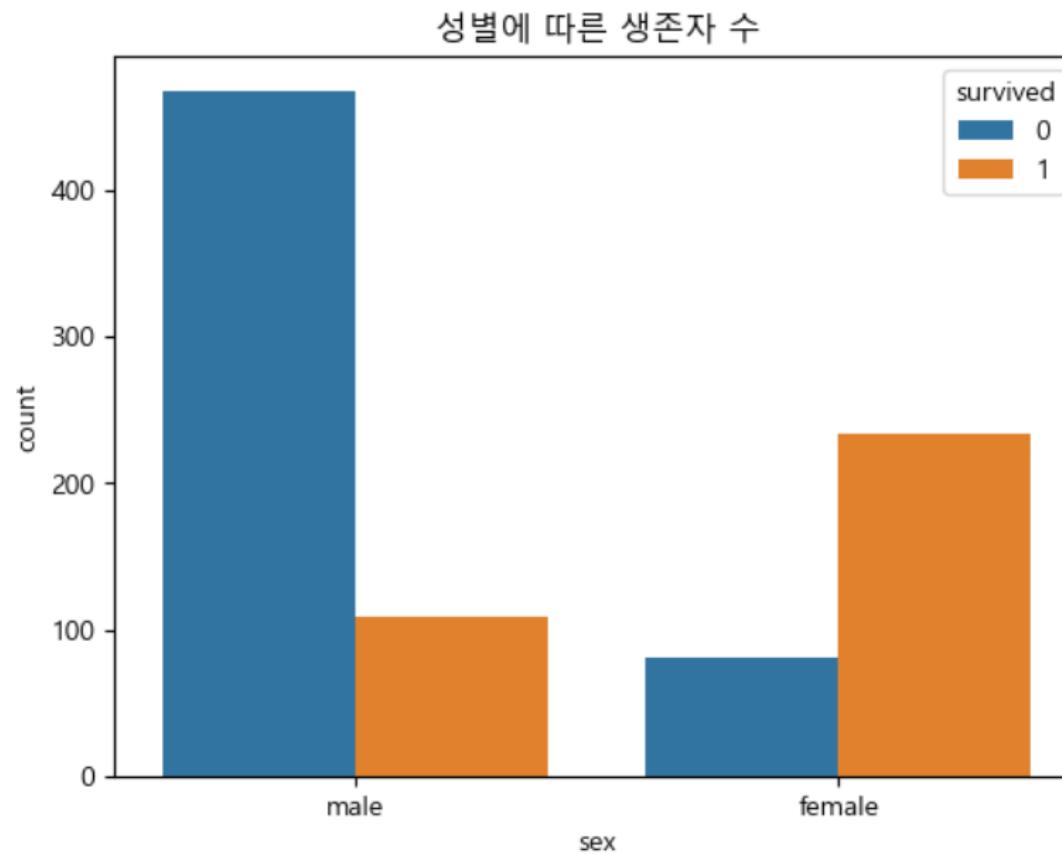
# 예제 데이터셋 로드
df = sns.load_dataset('titanic')

# 데이터 확인
print(df.head())

# 성별에 따른 생존자 수 시각화 (막대그래프)
sns.countplot(data=df, x='sex', hue='survived')
plt.title('성별에 따른 생존자 수')
plt.show()
```

# 타이타닉호 사례 분석

- 타이타닉호 - 성별에 따른 생존자 수 그래프



# 머신러닝(Machine Learning)

+ 머신러닝에 대해 설명해 줘

0



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



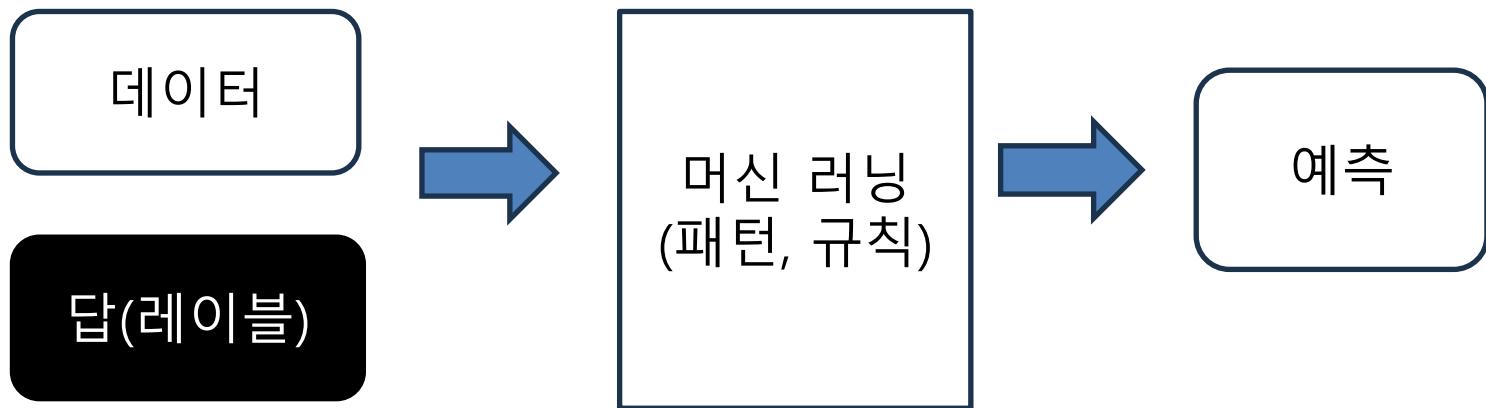
# 머신러닝(Machine Learning)

## ❖ 지도 학습 vs 비지도 학습

	지도 학습	비지도학습
분석 모형 (알고리즘)	<ul style="list-style-type: none"><li>회귀</li><li>분류</li></ul>	<ul style="list-style-type: none"><li>군집</li><li>차원 축소</li></ul>
특징	<ul style="list-style-type: none"><li>정답(레이블)을 알고 있는 상태에서 학습</li><li><b>회귀</b> – 가격, 매출, 주가 예측</li><li><b>분류</b> – 고객분류, 질병 진단, 스팸 메일 필터링</li></ul>	<ul style="list-style-type: none"><li>정답(레이블)이 없는 상태에서 서로 비슷한 데이터를 찾아서 그룹화(클러스터링)</li><li>구매 패턴 분석, 유튜브 추천 등</li></ul>

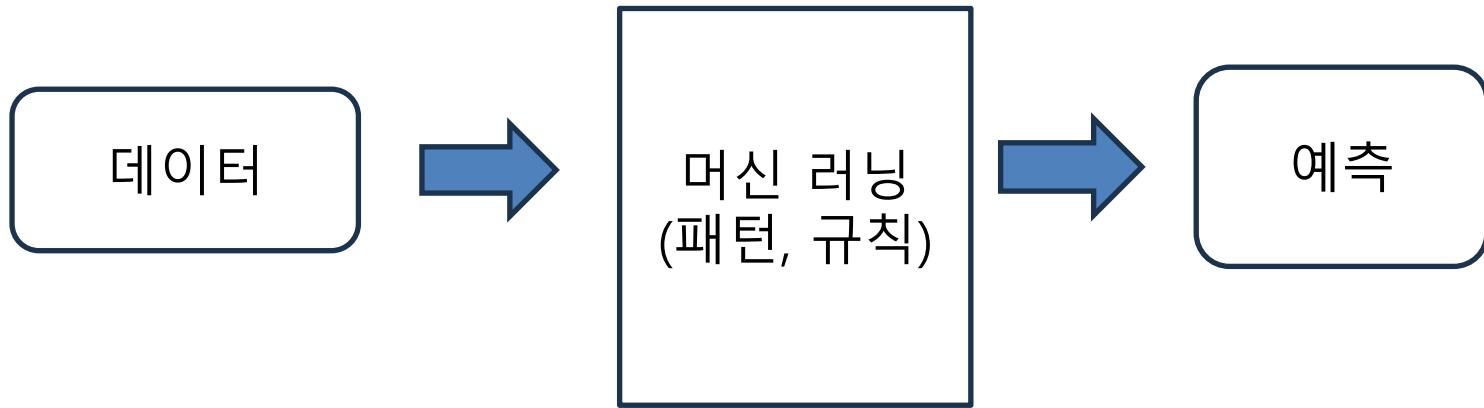
# 머신러닝(Machine Learning)

## ❖ 지도 학습



# 머신러닝(Machine Learning)

## ❖ 비지도 학습



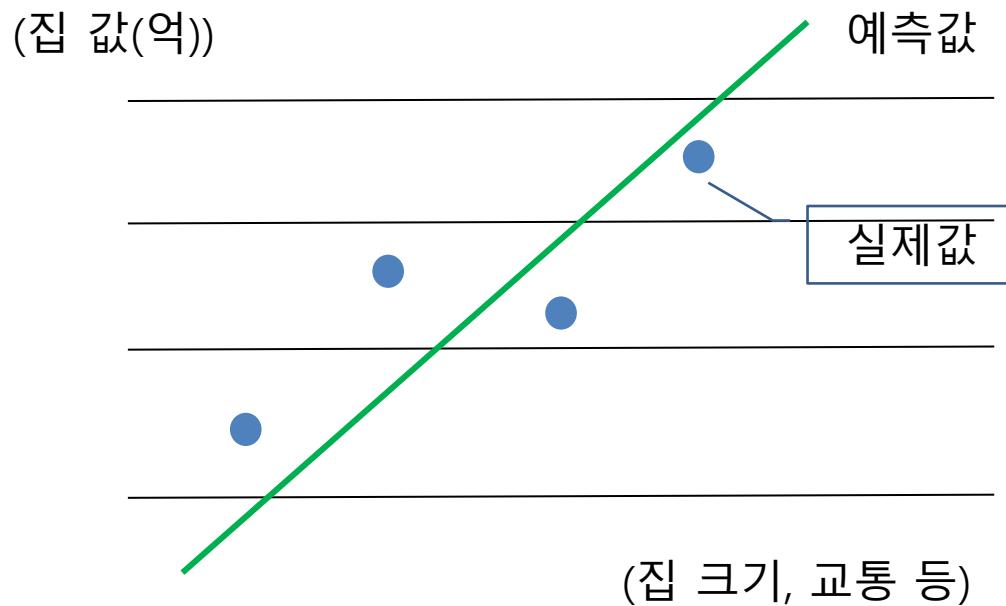
# 머신러닝(Machine Learning)

## ❖ 강화 학습

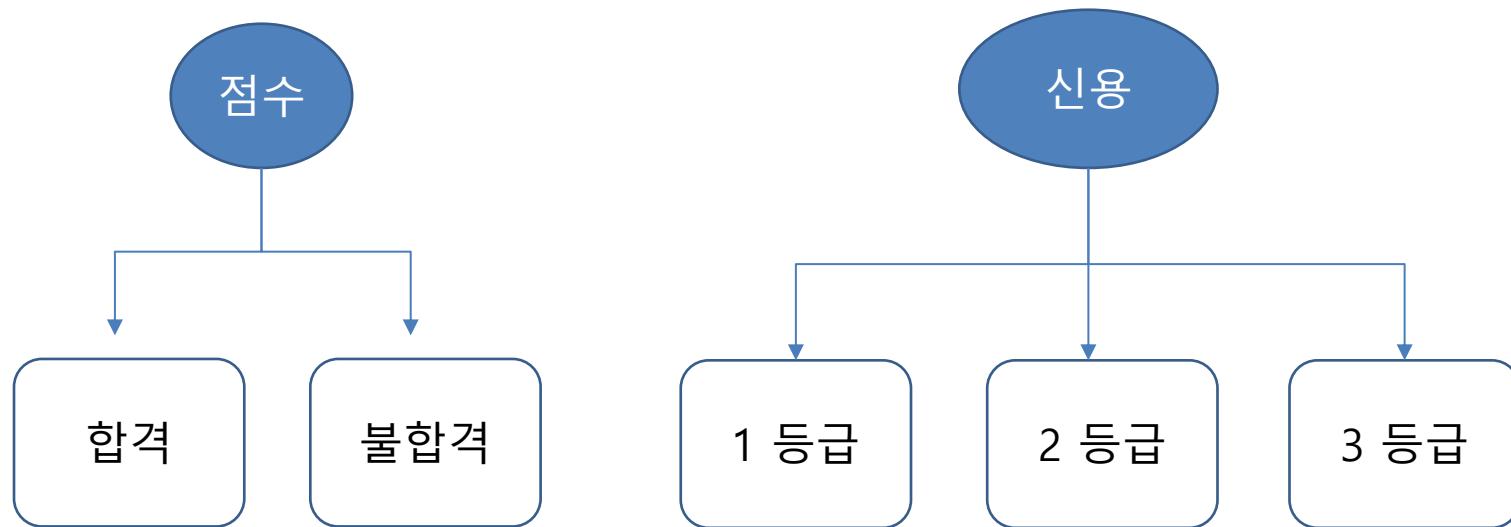
- 강화란 '어떤 것의 수준이나 정도를 높인다'는 의미로 인공지능의 수준을 높이는 것을 말한다.
  - 에이전트가 환경과 상호작용하면서 보상 신호에 따라 최적의 정책을 찾아내는 방법
- 
- 벽돌깨기 게임 – 딥마인드(구글) 회사
  - 알파고(딥마인드) – 바둑을 학습한 인공지능
  - 자율주행 자동차, 인공지능 로봇으로 확장되고 있음



# 회귀(Regression)

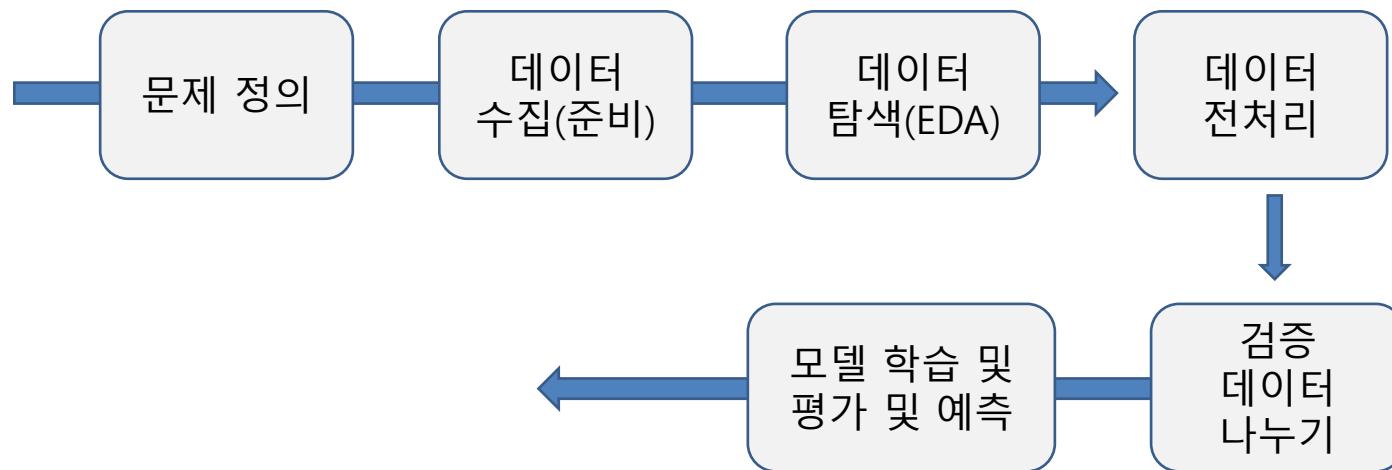


# 분류(Classification)



# 머신러닝 프로세스

- 머신러닝 모델 예측 프로세싱

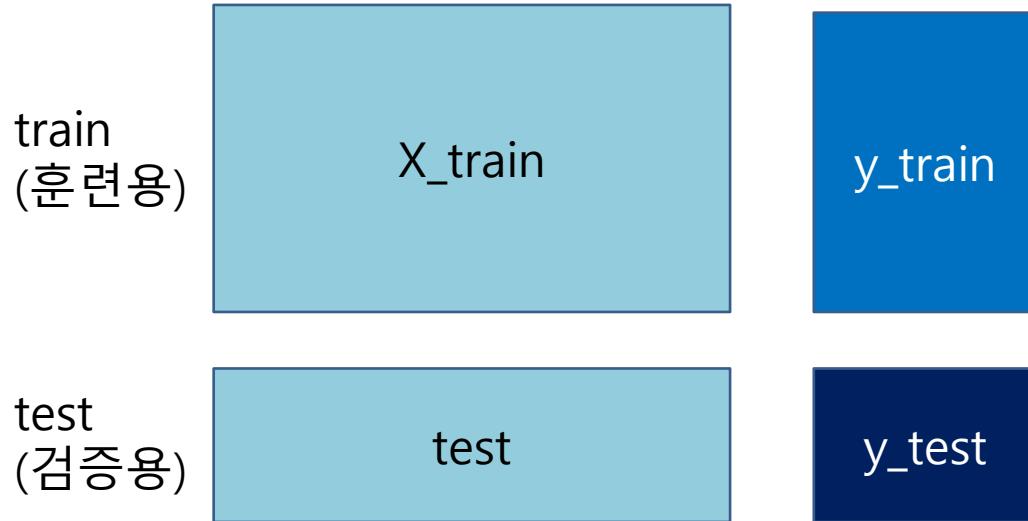


# 데이터 수집(준비)

- 데이터는 학습용과 검증으로 준비한다.

- ✓ 훈련 : 검증 비율

- 80 : 20 or 70 : 30



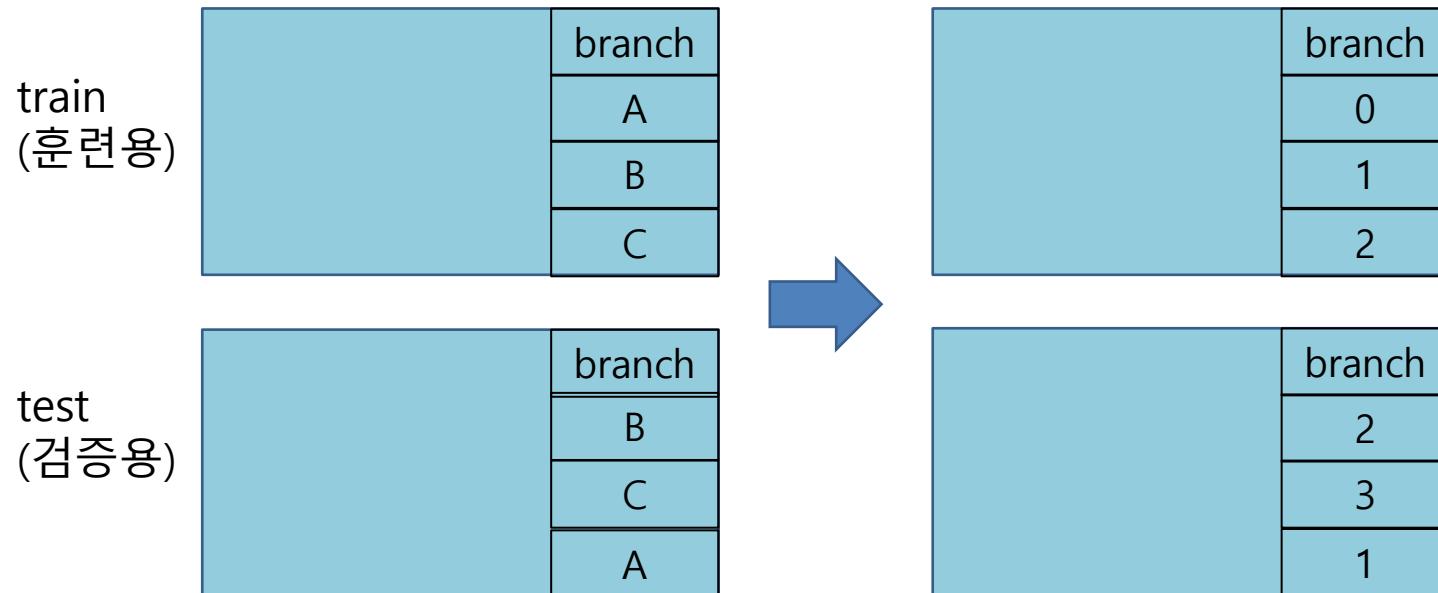
# 데이터 전처리

- 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

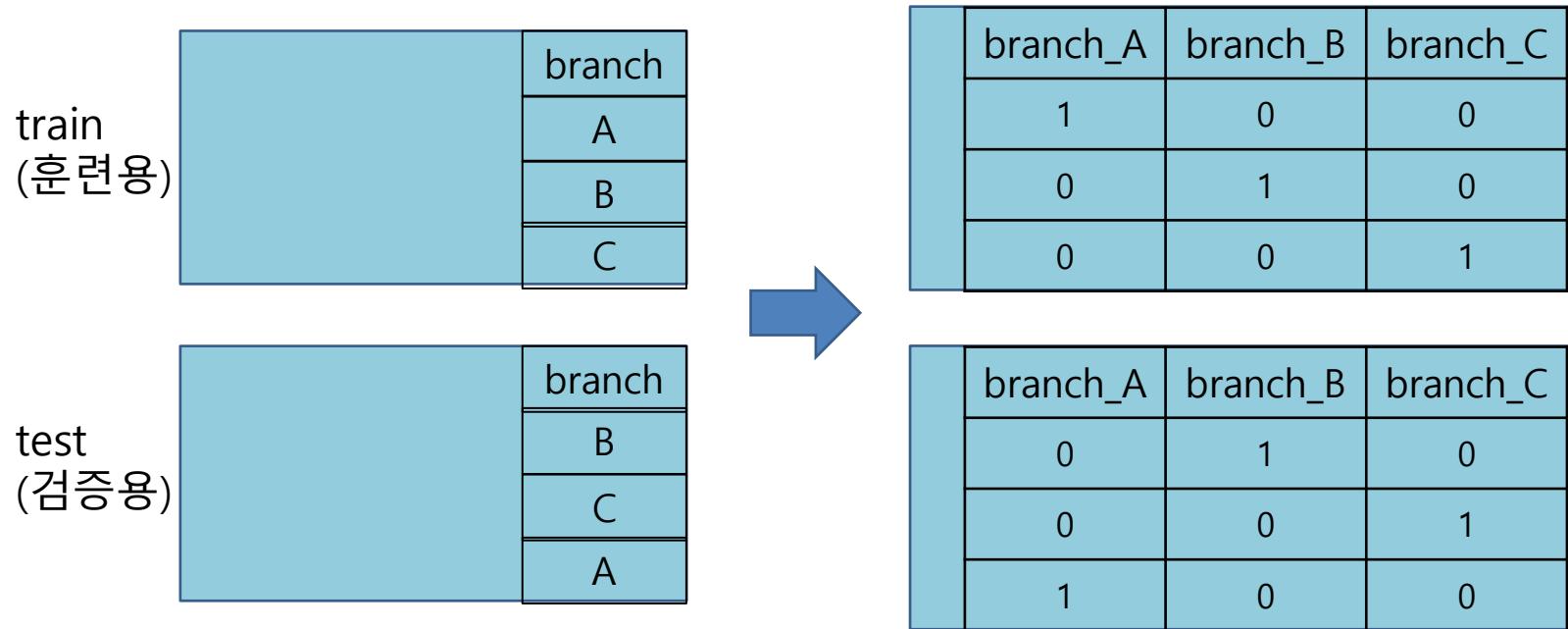
# 데이터 전처리

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



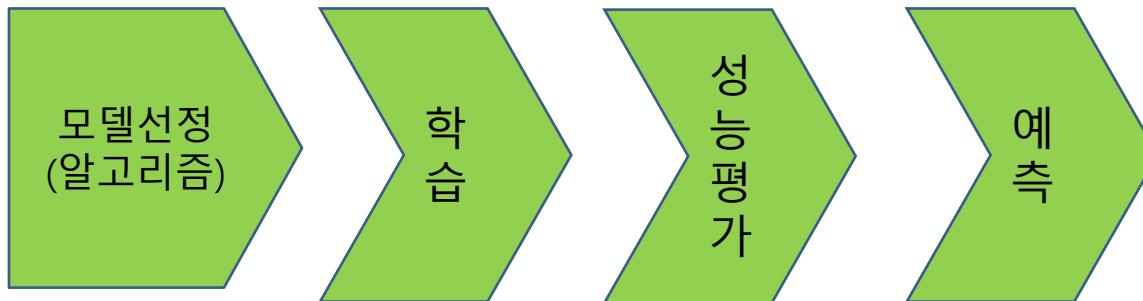
# 모델 학습 및 평가

## ● 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)

## ❖ 선형 회귀



선형 회귀에 대해서 설명해줘

- ◆ 선형 회귀(Linear Regression)는 가장 기본적이고 널리 사용되는 통계 및 머신러닝 기법 중 하나입니다. 한 개 이상의 독립 변수(설명 변수)와 종속 변수(예측 변수) 사이의 선형 관계를 모델링하는 데 사용됩니다. 즉, 입력 값(공부 시간)과 출력 값(시험 점수) 사이에 가장 잘 맞는 직선을 찾아 그 관계를 설명하고 미래 값을 예측하는 것이 목

공부 시간과 시험 점수 데이터를 사용하여 선형 회귀 모델을 만들어줘

선형 회귀의 수학적 원리에 대해 설명



빌드하는 데 어떤 도움이 필요하신가요?



Gemini 2.5 Flash ▾ ▶

Gemini는 실수할 수 있으므로, 대답을 다시 한번 확인하고 신중하게 코드를 사용하시기 바랍니다. [자세히 알아보기](#)



# 선형 회귀 분석

## ● 단순 선형 회귀

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

## ● 다중 선형 회귀

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



# 선형 회귀 분석

## ● 공부한 시간의 차이에 따른 시험 성적 예측하기

- 성적을 변하게 하는 요소를  $x$ 라 하고 이  $x$ 값에 따라 변하는 '성적'을  $y$ 라 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.

$$y = ax + b \text{(일차함수)}$$

기울기  $a$ 와 절편  $b$ 를 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 :  $(2 + 4 + 6 + 8) \div 4 = 5$

성적(y) 평균:  $(81 + 93 + 91 + 97) \div 4 = 90.5$

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 stdDev라고 함.

# 최소 제곱법

- 공부한 시간의 차이에 따른 시험 성적 예측하기

- y절편 b값 구하기

$$b = y\text{의 평균} - (\text{기울기 } a \times (\text{x의 평균}))$$

$$b = 90.5 - (2.3 \times 5) = 79$$

최적의 직선(방정식)

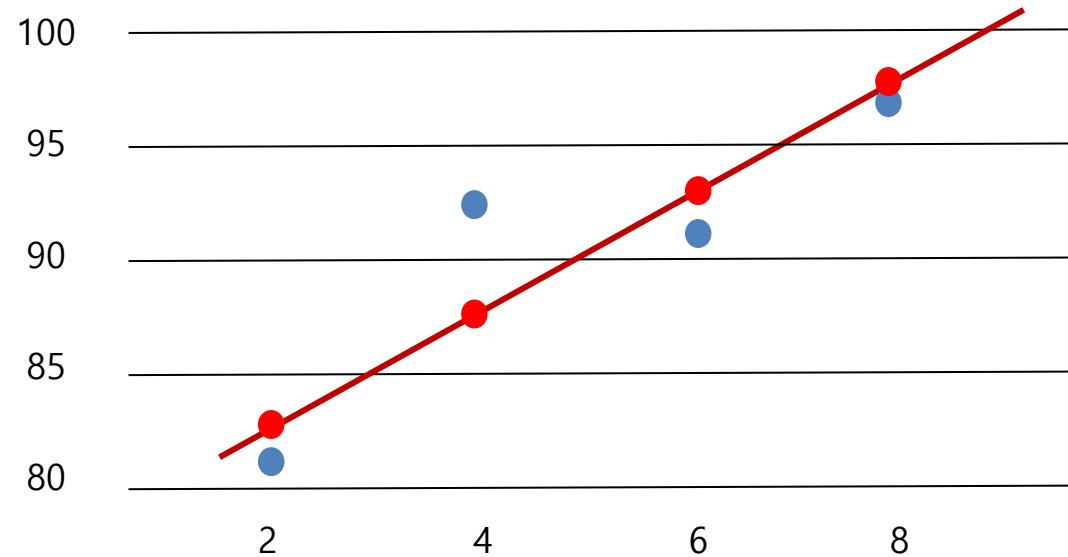
$$y = 2.3x + 79$$

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

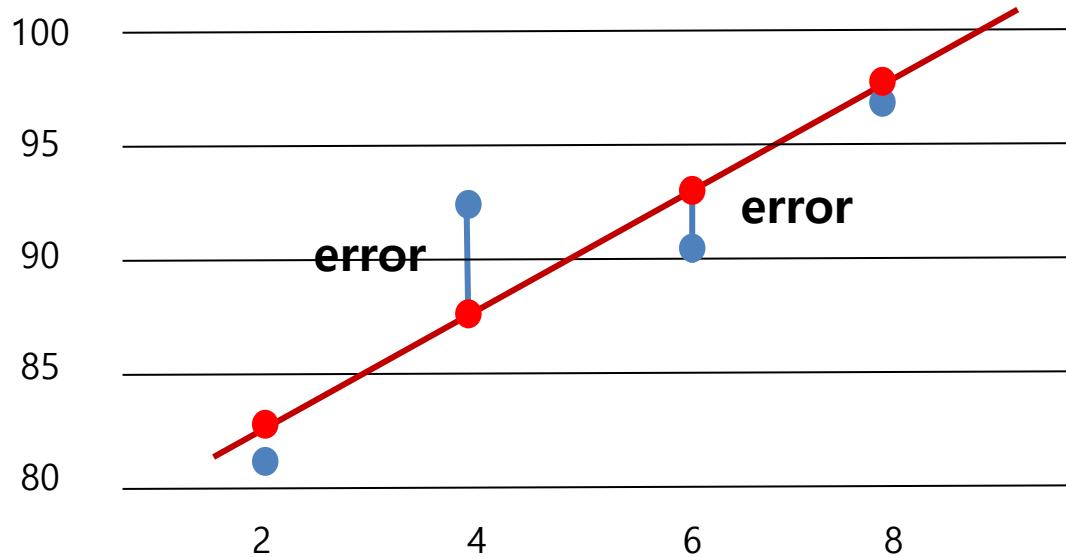
공부 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4



# 평균 제곱 오차

## ▪ 평균 제곱 오차

선형 회귀는 임의의 직선을 그어 이에 대한 평균 제곱 오차를 구하고, 이를 값은 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

$$\text{분모} : 1 + 25 + 9 + 9 = 44$$

$$\text{MSE} = 44 / 4 = 11$$

$$\text{MAE} = 1+5+3+3 = 12$$

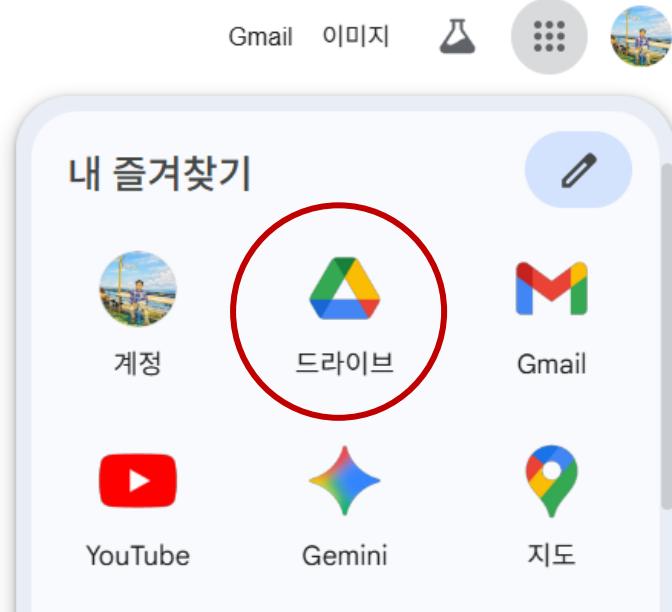


# 구글 코렙

## ❖ Google Colab 사용하기

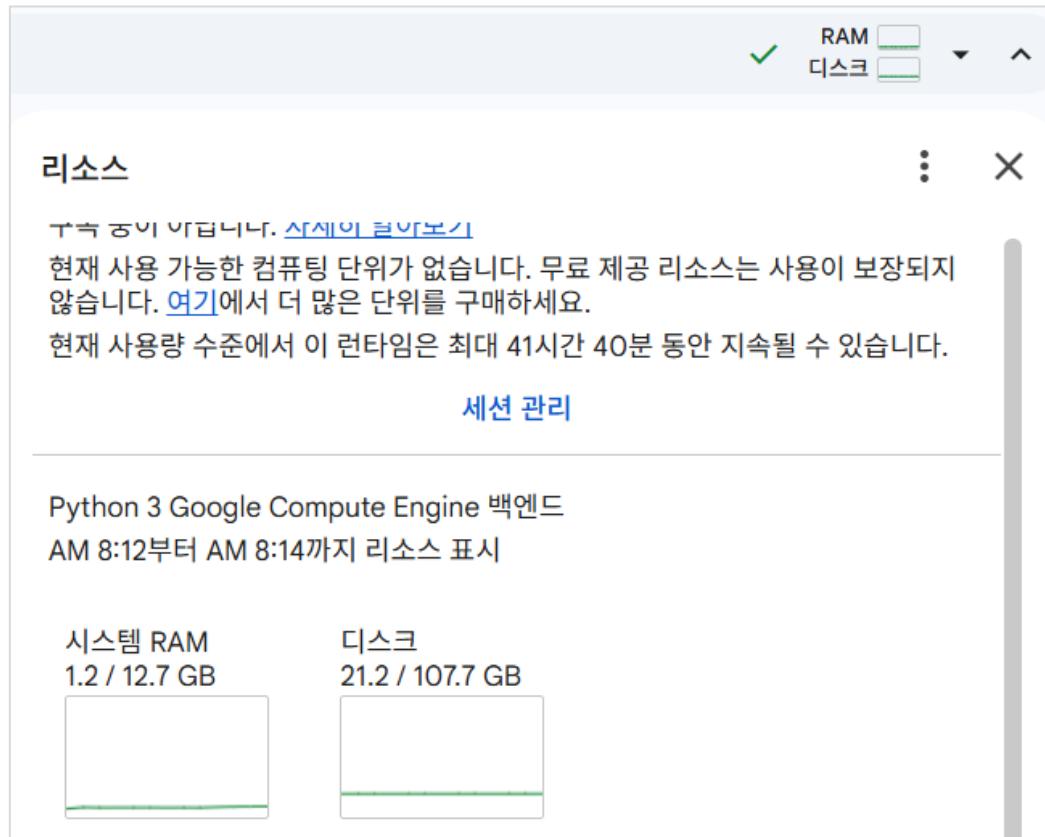
Colaboratory는 구글의 Jupyter Notebook 기반 클라우드 서비스로, TPU나 GPU, CPU를 무료 제공하며 클라우드 IDE의 일부이다

- 계정 로그인 > 메뉴(선택) > 드라이브



# 구글 코렙

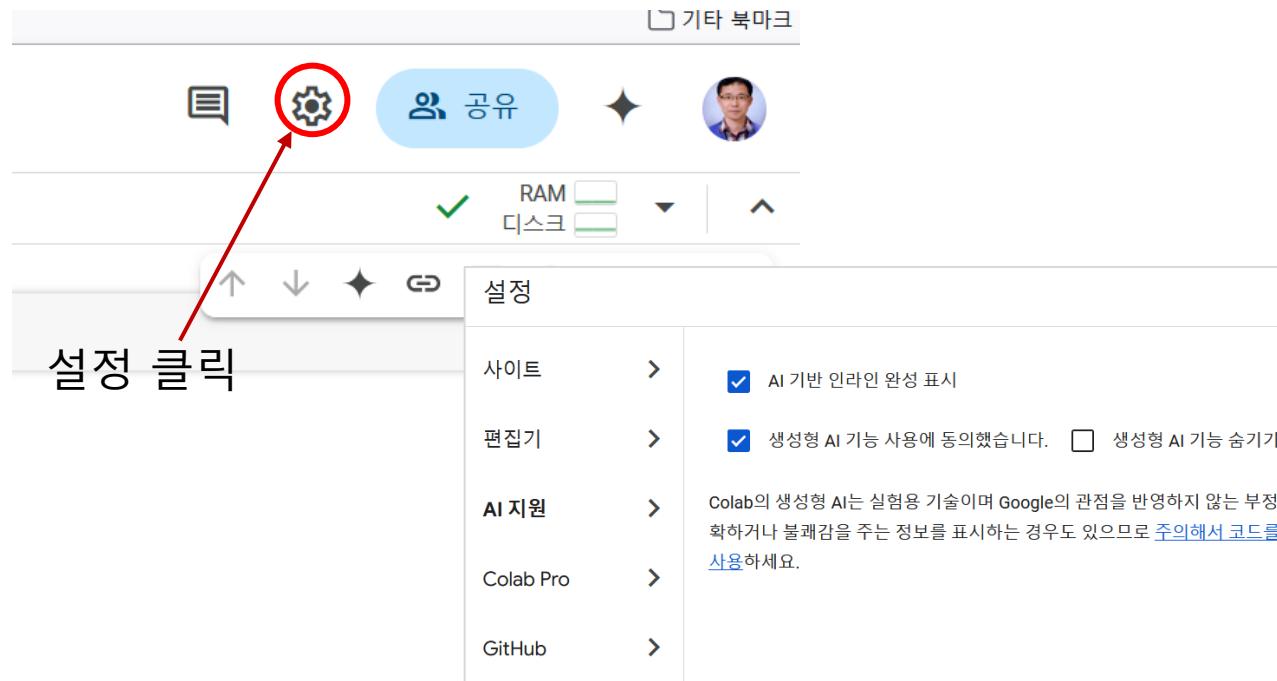
## ❖ 리소스 사용 - 클라우드 컴퓨터 사용



# 구글 코랩

## ❖ 구글 코랩(Colaboratory)

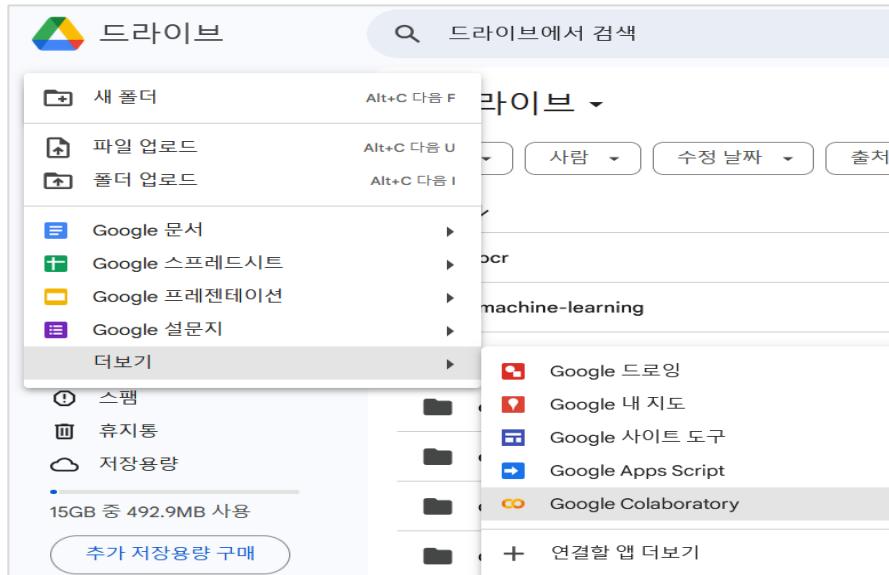
AI(Gemini) 지원 : 설정 > AI 지원 > 동의



# 구글 코랩

## ❖ 새 노트 만들기

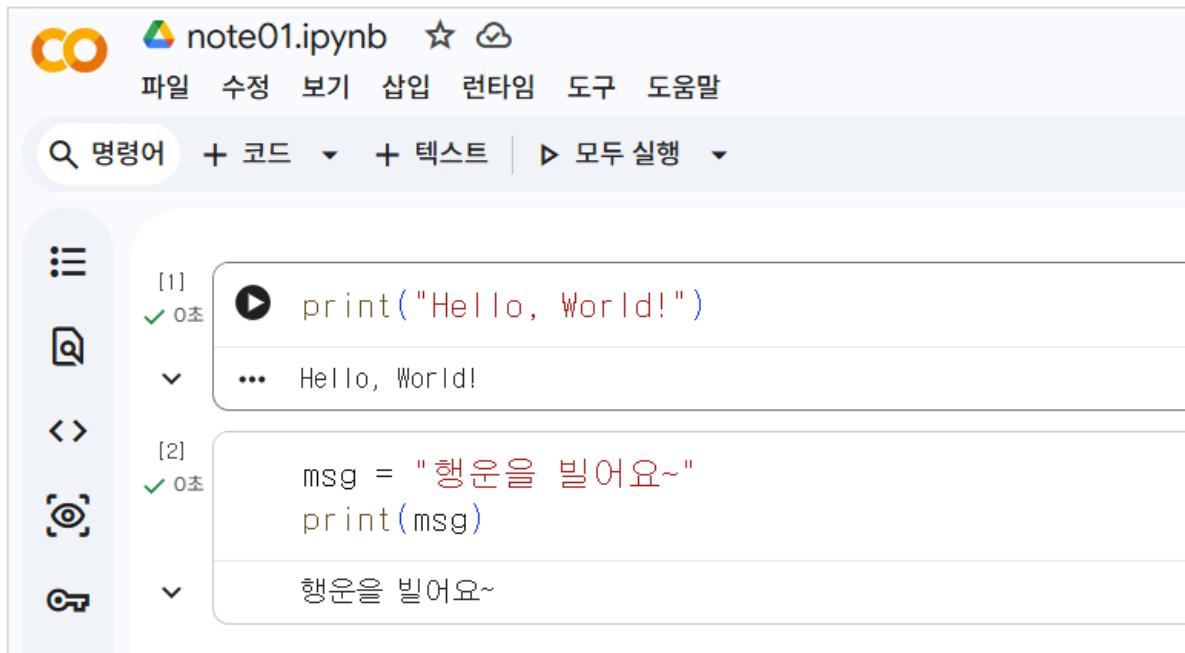
- 드라이브 > 홈
- + 신규 > 더보기 > Google Colaboratory



# 구글 코랩

## ❖ 새 노트 만들기

- 파일이름 > note01.ipynb



```
[1]: print("Hello, World!")
... Hello, World!

[2]: msg = "행운을 빌어요~"
print(msg)
... 행운을 빌어요~
```

# 구글 코렙

## ❖ 새 노트 만들기

- 파일이름 > note01.ipynb

The screenshot shows the Google Colab interface. At the top, there is a play button icon followed by Python code:

```
import pandas as pd
```

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

df

Below the code, the resulting data frame is displayed:

	name	price
0	마우스	20000
1	모니터	80000

At the bottom, there is a message in Korean: "다음 단계: df 변수로 코드 생성" (Next step: Create code using variable df) and a button labeled "New interactive sheet".

# 구글 코렙

## ❖ 새 폴더 만들기

- 드라이브 > 내 드라이브
- + 신규 > 새폴더> 머신러닝

내 드라이브 ▾

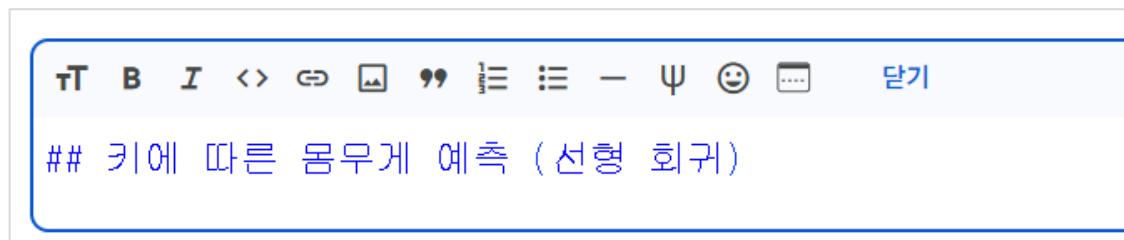
유형 ▾ 사람 ▾ 수정 날짜 ▾ 출처 ▾

이름	▲	소유자
머신러닝		나
note01.ipynb		나

# 구글 코랩

## ❖ 텍스트 사용하기

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



<편집화면>

shift + enter

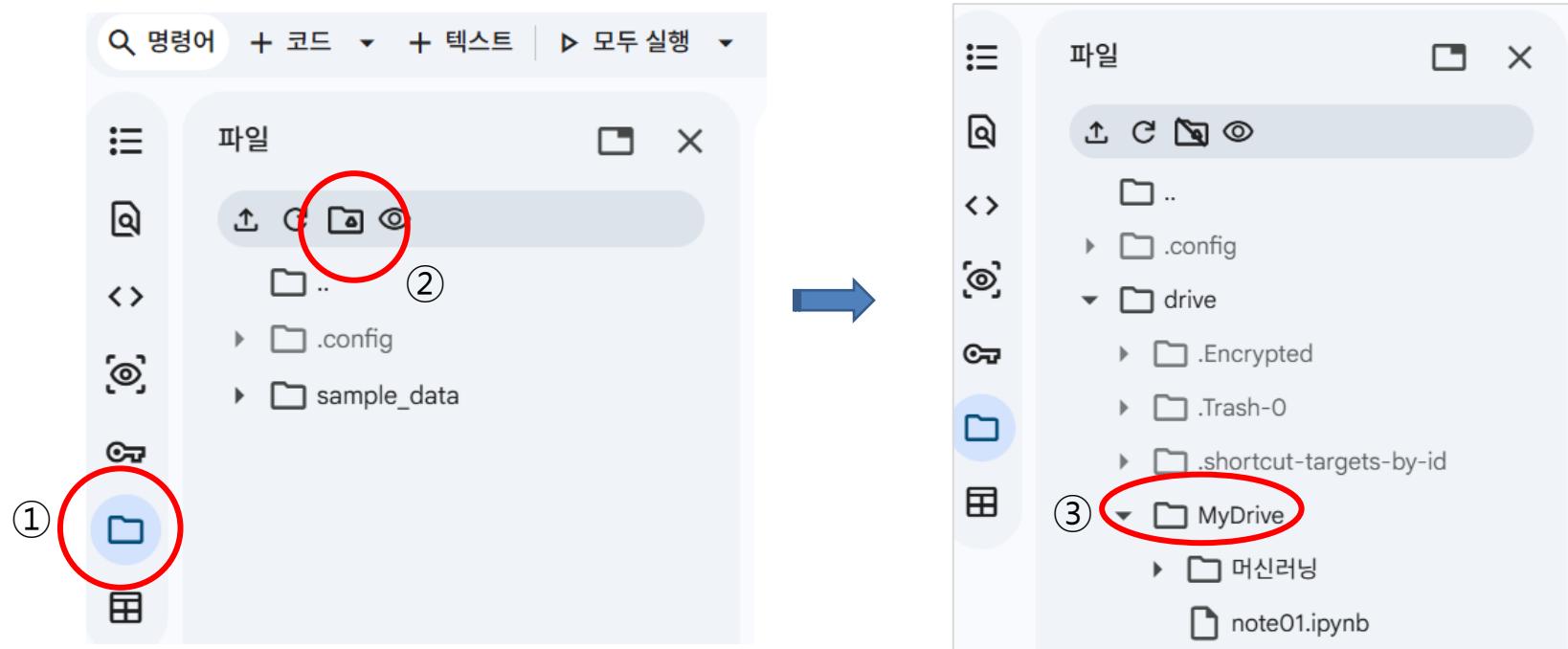
키에 따른 몸무게 예측 (선형 회귀)

<출력화면>

# 구글 코렙

## ❖ 드라이브 마운트

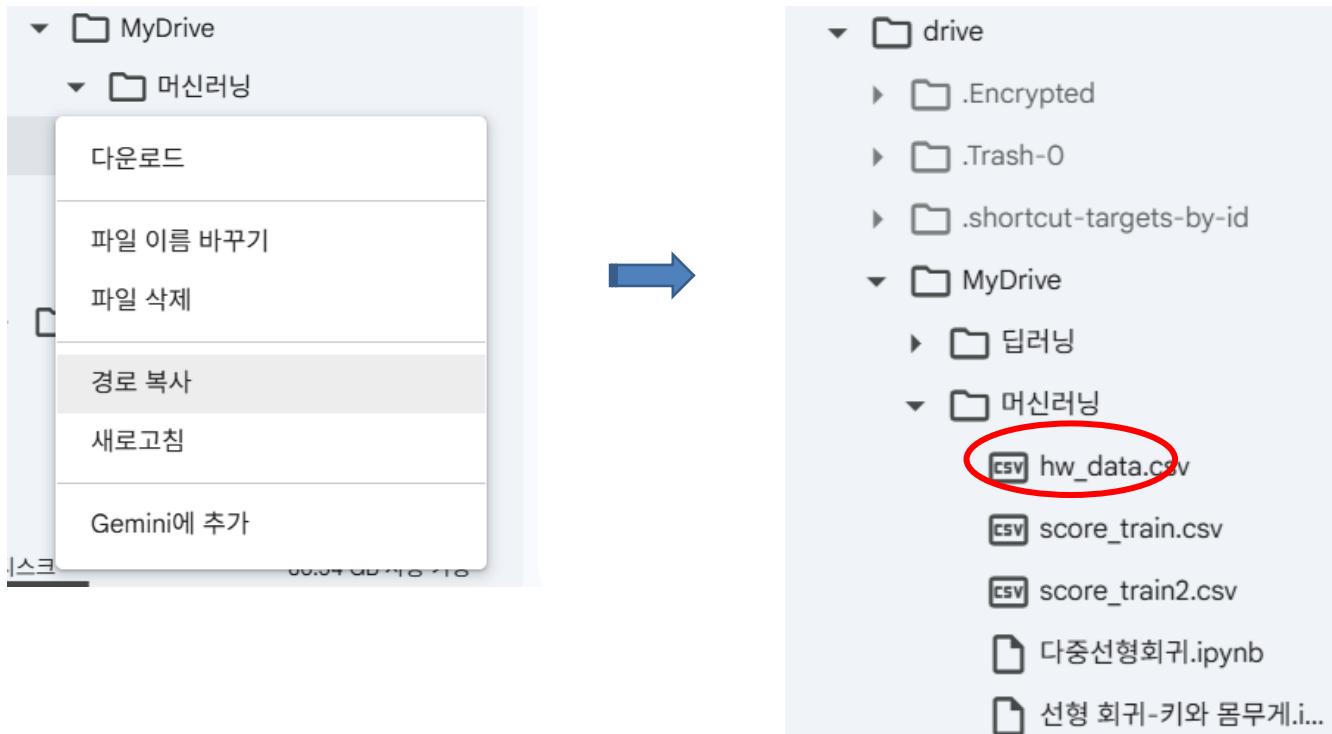
- ▶ 폴더 > 드라이브 마운트 > 드라이브 연결



# 구글 코렙

## ❖ 훈련 데이터 업로드

- (...) 메뉴 > 업로드 > data\_hw.csv



# 키에 따른 몸무게 예측

## ● 데이터 준비

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# 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/머신러닝/hw_data.csv", index=False)

df_hw.head()
```

# 키에 따른 몸무게 예측

## ● 변수 선택

### 변수 선택

- 독립변수 (X): 키
- 종속변수 (y): 몸무게

```
▶ # csv 파일 읽기  
df_hw = pd.read_csv("/content/drive/MyDrive/머신러닝/hw_data.csv")  
print(df_hw)  
  
# 독립 변수 '키'  
X_hw = df_hw[['키']]  
# 종속 변수 '몸무게'  
y_hw = df_hw['몸무게']  
  
print("X (키):\n", X_hw.head())  
print("y (몸무게):\n", y_hw.head())
```

# 키에 따른 몸무게 예측

## ● 모델 학습

### 모델 학습

```
# 선형 회귀 모델 초기화  
model_hw = LinearRegression()  
  
# 모델 학습  
model_hw.fit(X_hw, y_hw)  
  
print("키-몸무게 모델 학습이 완료되었습니다.")
```

키-몸무게 모델 학습이 완료되었습니다.

# 키에 따른 몸무게 예측

## ● 모델 학습 및 예측 수행

### ▼ 모델 학습

[3]  
✓ 0초

```
# 선형 회귀 모델 초기화  
model_hw = LinearRegression()  
  
# 모델 학습  
model_hw.fit(X_hw, y_hw)  
  
print("키-몸무게 모델 학습이 완료되었습니다.")
```

▼ 키-몸무게 모델 학습이 완료되었습니다.

### ▼ 예측 수행

[4]  
✓ 0초

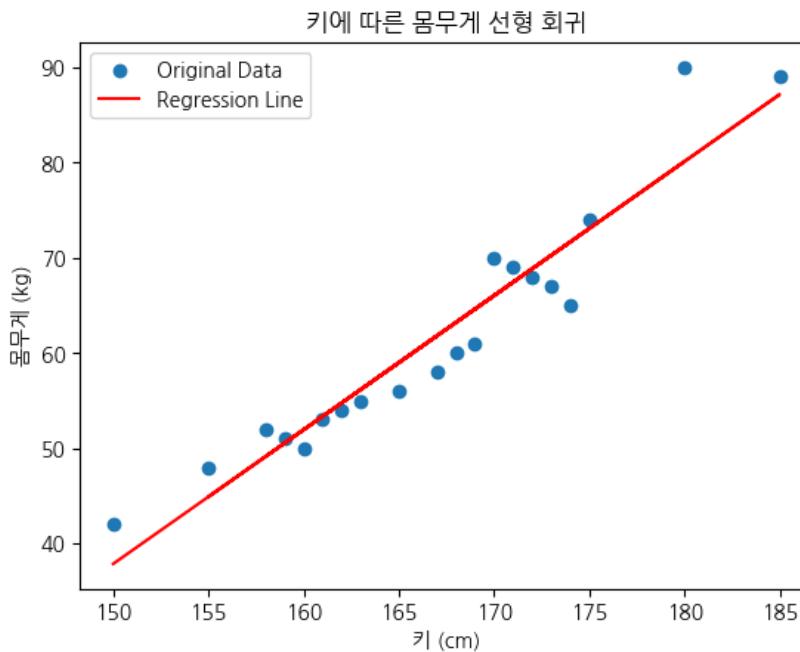
```
pred_hw = model_hw.predict(X_hw)  
  
print("예측 결과 (일부):\n", pred_hw[:5])
```

▼ 예측 결과 (일부):  
[37.87287743 51.95425581 66.03563419 73.07632338 58.994945 ]

# 키에 따른 몸무게 예측

## ● 모델 시각화

```
plt.scatter(X_hw, y_hw, label='Original Data')
plt.plot(X_hw, pred_hw, 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

## ● 성능 평가 및 몸무게 예측

직선의 방정식 (기울기 및 y절편)

```
print(f"기울기 (Coefficient): {model_hw.coef_[0]:.4f}")
print(f"절편 (Intercept): {model_hw.intercept_:.4f}")
```

```
기울기 (Coefficient): 1.4081
절편 (Intercept): -173.3478
```

성능 평가

```
# 키-몸무게 모델 (model_hw)에 대한 성능 평가 (전체 데이터)
```

```
print(f"키-몸무게 모델 (전체 데이터) R2 Score: {model_hw.score(X_hw, y_hw):.4f}")
```

```
키-몸무게 모델 (전체 데이터) R2 Score: 0.9135
```

키가 170cm일 때 예상 몸무게

```
print(model_hw.predict([[170]]))
```

```
[66.03563419]
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names.
  warnings.warn(
```

# 키에 따른 몸무게 예측

## ● 데이터 세트 분리

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# csv 파일 읽기
df_hw = pd.read_csv("/content/drive/MyDrive/머신러닝/hw_data.csv")
print(df_hw.head())

# 변수 선택
# X = df_hw[['키']]
X = df_hw.drop("몸무게", axis=1)
y = df_hw['몸무게']

print("X (키):\n", X.head())
print("y (몸무게):\n", y.head())
```

	키	몸무게
0	150	42
1	160	50
2	170	70
3	175	65
4	165	60

X (키):

	키
0	150
1	160
2	170
3	175
4	165

# 키에 따른 몸무게 예측

## ● 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=0.2, random_state=12) # 테스트 세트 크기를 20%로 변경

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

# 훈련 세트와 테스트 세트 확인
print("훈련 세트 X_train:\n", X_train.head())
print("\n훈련 세트 y_train:\n", y_train.head())
print("\n테스트 세트 X_test:\n", X_test.head())
print("\n테스트 세트 y_test:\n", y_test.head())

(16, 1) (4, 1) (16,) (4,)
훈련 세트 X_train:
    몸무게
14    58
16    69
10    54
8     65
5     48

훈련 세트 y_train:
    14    58
    16    69
    10    54
    8     65
    5     48
Name: 몸무게, dtype: int64
```

# 키에 따른 몸무게 예측

## ● 모델 학습

분리된 데이터를 통한 모델 학습

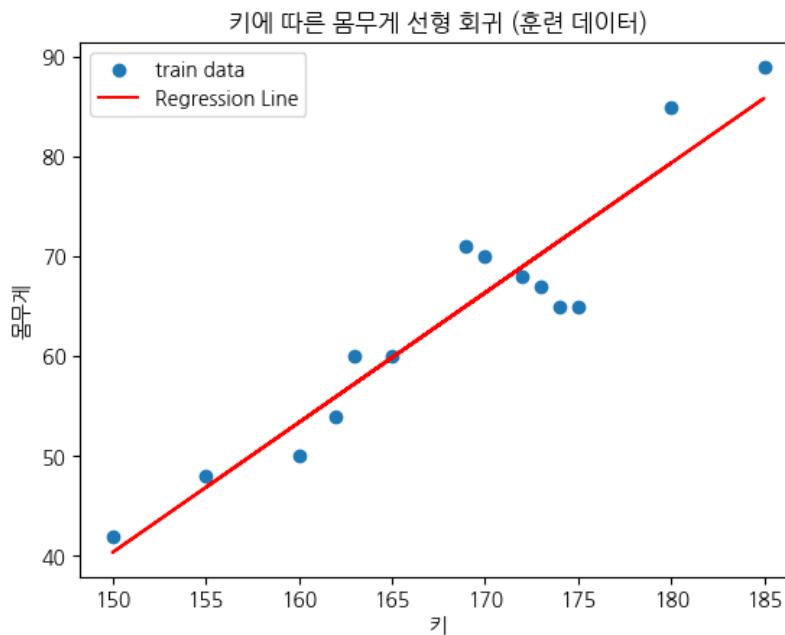
```
from sklearn.linear_model import LinearRegression  
  
# 선형 회귀 모델 객체 생성  
model = LinearRegression()  
  
# 모델 학습  
model.fit(X_train, y_train)  
  
print("모델 학습이 완료되었습니다.")
```

모델 학습이 완료되었습니다.

# 키에 따른 몸무게 예측

## ● 모델 시각화 – 훈련 데이터

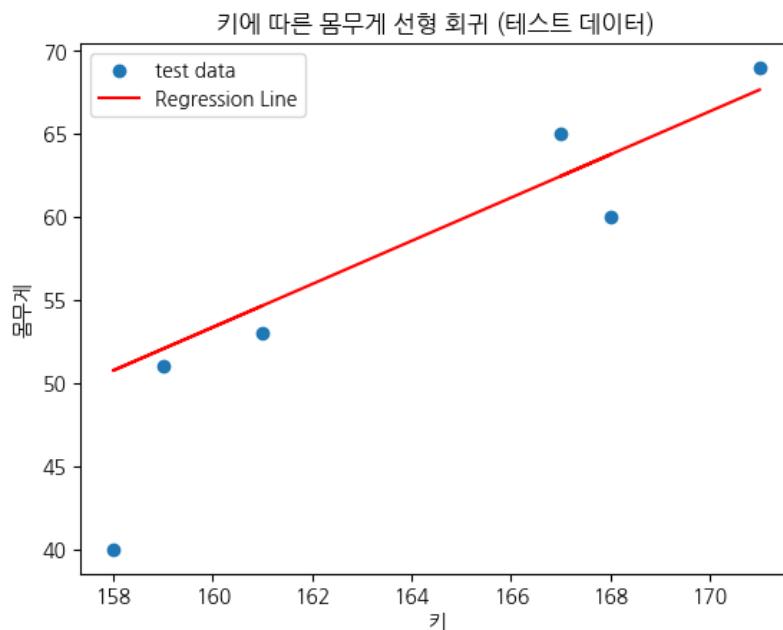
```
plt.scatter(X_train, y_train, label='train data')
plt.plot(X_train, model.predict(X_train), color='red', label='Regression Line')
plt.rc('font', family='NanumGothic') # 한글 글꼴 적용
plt.title('키에 따른 몸무게 선형 회귀 (훈련 데이터)')
plt.xlabel('키')
plt.ylabel('몸무게')
plt.legend()
plt.show()
```



# 키에 따른 몸무게 예측

## ● 모델 시각화 – 테스트 데이터

```
plt.scatter(X_test, y_test, label='test data') # 레이블 설정  
plt.plot(X_test, model.predict(X_test), color='red', label='Regression Line')  
plt.rc('font', family='NanumGothic') # 한글 글꼴 적용  
plt.title('키에 따른 몸무게 선형 회귀 (테스트 데이터)') # 제목 설정  
plt.xlabel('키') # 라벨 설정  
plt.ylabel('몸무게') # 라벨 설정  
plt.legend()  
plt.show()
```



# 키에 따른 몸무게 예측

## ● 모델 성능 평가 및 예측

### 모델의 성능 평가

```
model.score(X_train, y_train) #훈련세트를 통한 모델 평가
```

```
0.896152355234782
```

```
model.score(X_test, y_test) #테스트 세트를 통한 모델 평가
```

```
0.7439908761659375
```

### 키가 170cm일때 예상 몸무게

```
print(model.predict([[170]]))
```

```
[66.36418161]
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names - [
```

```
warnings.warn(
```

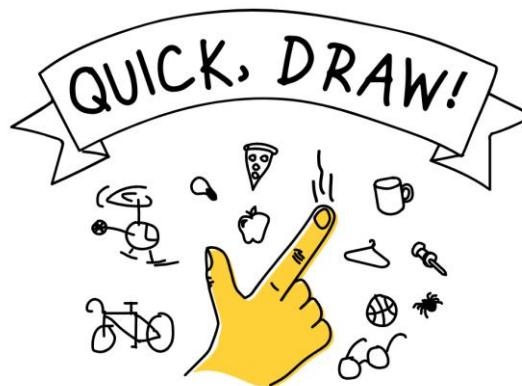
# 딥러닝(Deep Learning)

## ● 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.

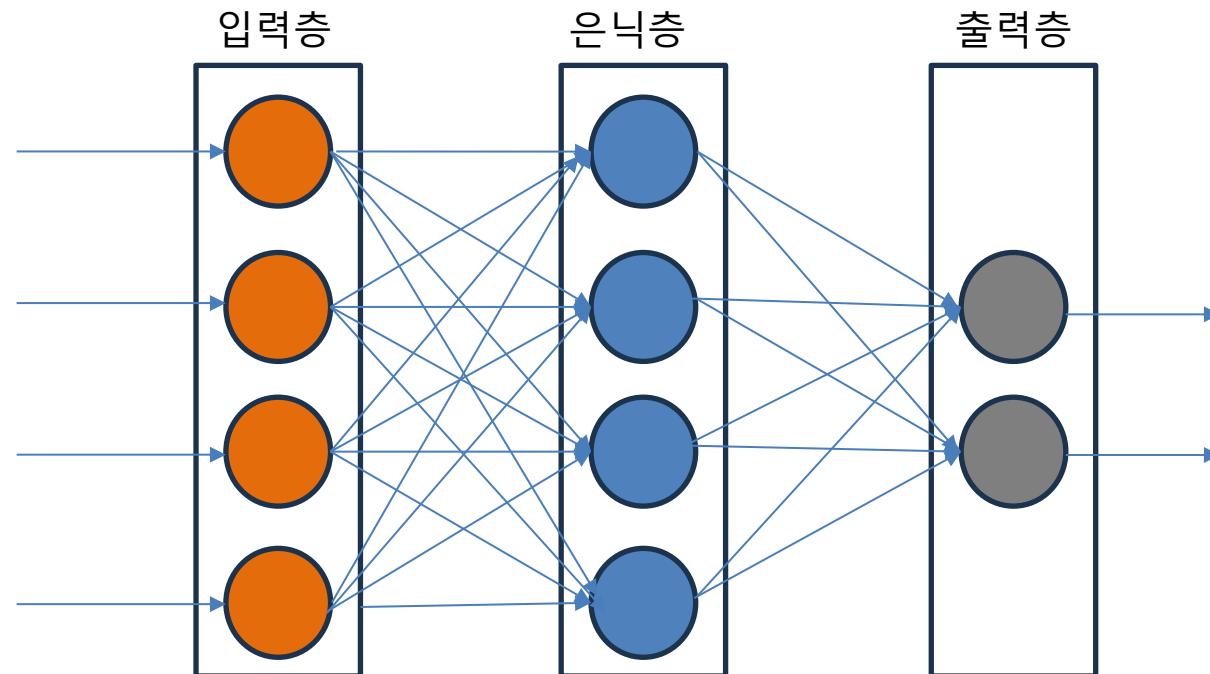
입력층 -> 은닉층 -> 은닉층 -> 출력층



# 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 전달**합니다.

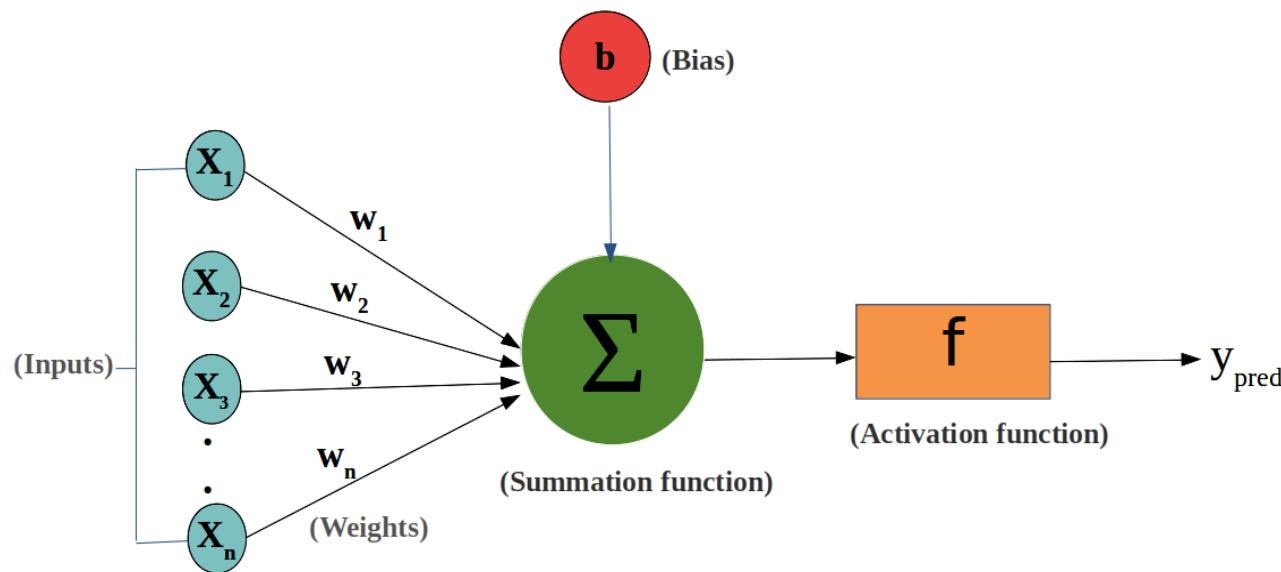
신경망에서 중요한 용어 중 **가중치(weight)**와 **편향(bias)**이 있는데 이들이 바로 신호 세기를 변경하는 데 사용된다.

한 뉴런에서 다음 뉴런으로 전달되는 신호 세기는 가중치와 편향에 의해 결정됨

**가중치**란 그 값이 얼마나 중요한지 그렇지 않은지를 표현하기 위한 도구이고, **편향**은 한쪽으로 치우치는 값을 더할 때 사용함

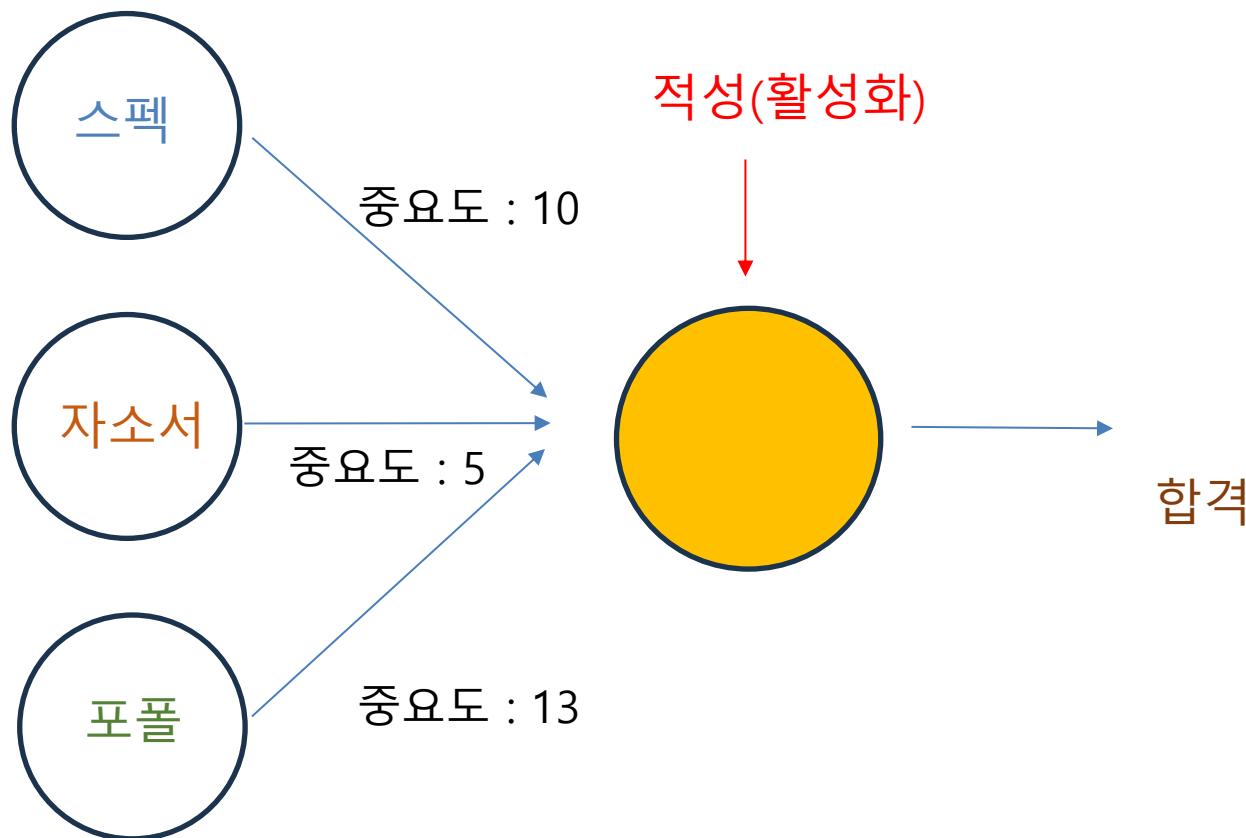
# 딥러닝(Deep Learning)

- 가중치와 편향



# 딥러닝(Deep Learning)

- 가중치와 편향



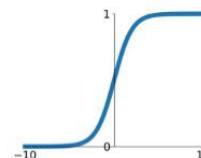
# 딥러닝(Deep Learning)

## ● 활성화 함수

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

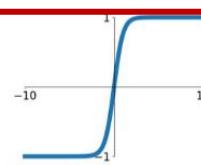
**Sigmoid**

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



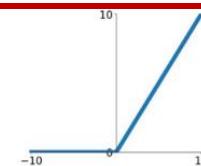
**tanh**

$$\tanh(x)$$



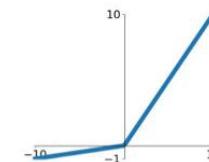
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

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

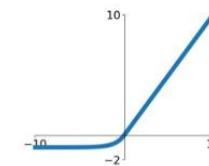


**Maxout**

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

**ELU**

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



# 딥러닝(Deep Learning)

## ● Softmax(소프트맥스) 함수

소프트 맥스 함수는 사실 활성화 함수는 아니고, 인공신경망의 마지막 부분 출력층에서 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 데이터셋의 이미지 하나를 시각화해줘

이미지 데이터를 신경망에 맞게 전처리해줘

첫 번째 딥러닝 모델을 만들어줘

MNIST에 대해 설명해 줘

+

Gemini 2.5 Flash ▾ ▶

### 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.

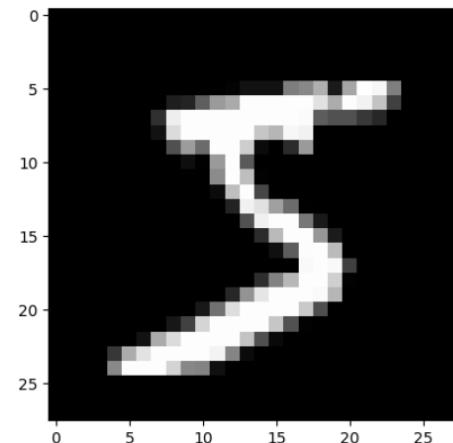
A 4x4 grid of handwritten digits from the MNIST dataset. The digits are arranged in a 4x4 pattern. Each digit is a black outline on a white background. The digits represent the numbers 0 through 9.

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9

# 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 이미지가 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

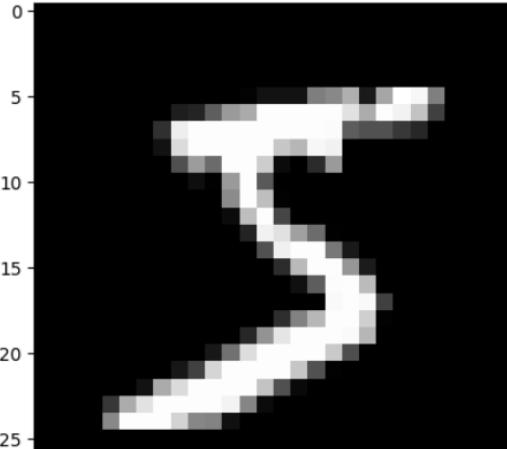
# 데이터 준비
(X_train, y_train), (X_test, y_test) = mnist.load_data()

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

0~255(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 바꿔

```
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 / 255
```

# 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.	, 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.	,	,
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.	, 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.	,	,
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.	, 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.	,	,

# 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

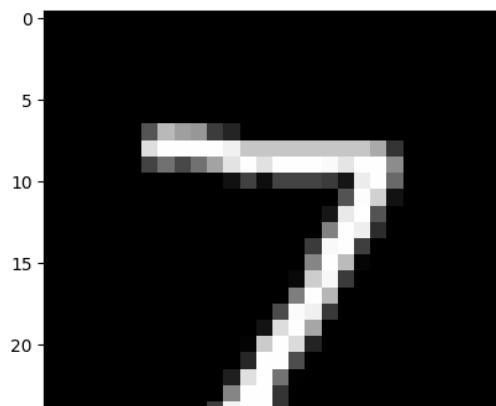
```
# 모델 컴파일(손실함수(오차) - 범주형: categorical_crossentropy)
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# 학습
model.fit(X_train, y_train, epochs=5, batch_size=100)
```

```
Epoch 1/5
600/600 [=====] 7s 9ms/step - accuracy: 0.9894 - loss: 0.0317
Epoch 2/5
600/600 [=====] 4s 7ms/step - accuracy: 0.9929 - loss: 0.0221
Epoch 3/5
600/600 [=====] 5s 8ms/step - accuracy: 0.9945 - loss: 0.0162
Epoch 4/5
600/600 [=====] 5s 8ms/step - accuracy: 0.9945 - loss: 0.0170
Epoch 5/5
600/600 [=====] 4s 7ms/step - accuracy: 0.9955 - loss: 0.0141
<keras.src.callbacks.history.History at 0x7d006a9b7a40>
```

```
plt.imshow(X_test[0].reshape(28, 28), cmap="gray") # 숫자를 이미지로 출력
```

```
<matplotlib.image.AxesImage at 0x7d000c7f1f10>
```



# 딥러닝(Deep Learning)

## ● 모델 예측

```
# 예측 - 0번째 숫자 예측하기  
model.predict(X_test[0].reshape(1, 784))  
  
1/1 ━━━━━━━━━━ 0s 38ms/step  
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)  
  
print(np.argmax(model.predict(X_test[0].reshape(1, 784))))  
  
1/1 ━━━━━━━━━━ 0s 34ms/step  
7
```