



# Artificial Intelligence Tutorial

박 경 규

# 강사소개



- 이름 : 박경규
- 강의경력: 딥러닝, 파이썬, 에너지솔루션
- 깃허브 주소 : <https://github.com/kgpark88>
- 개발경력

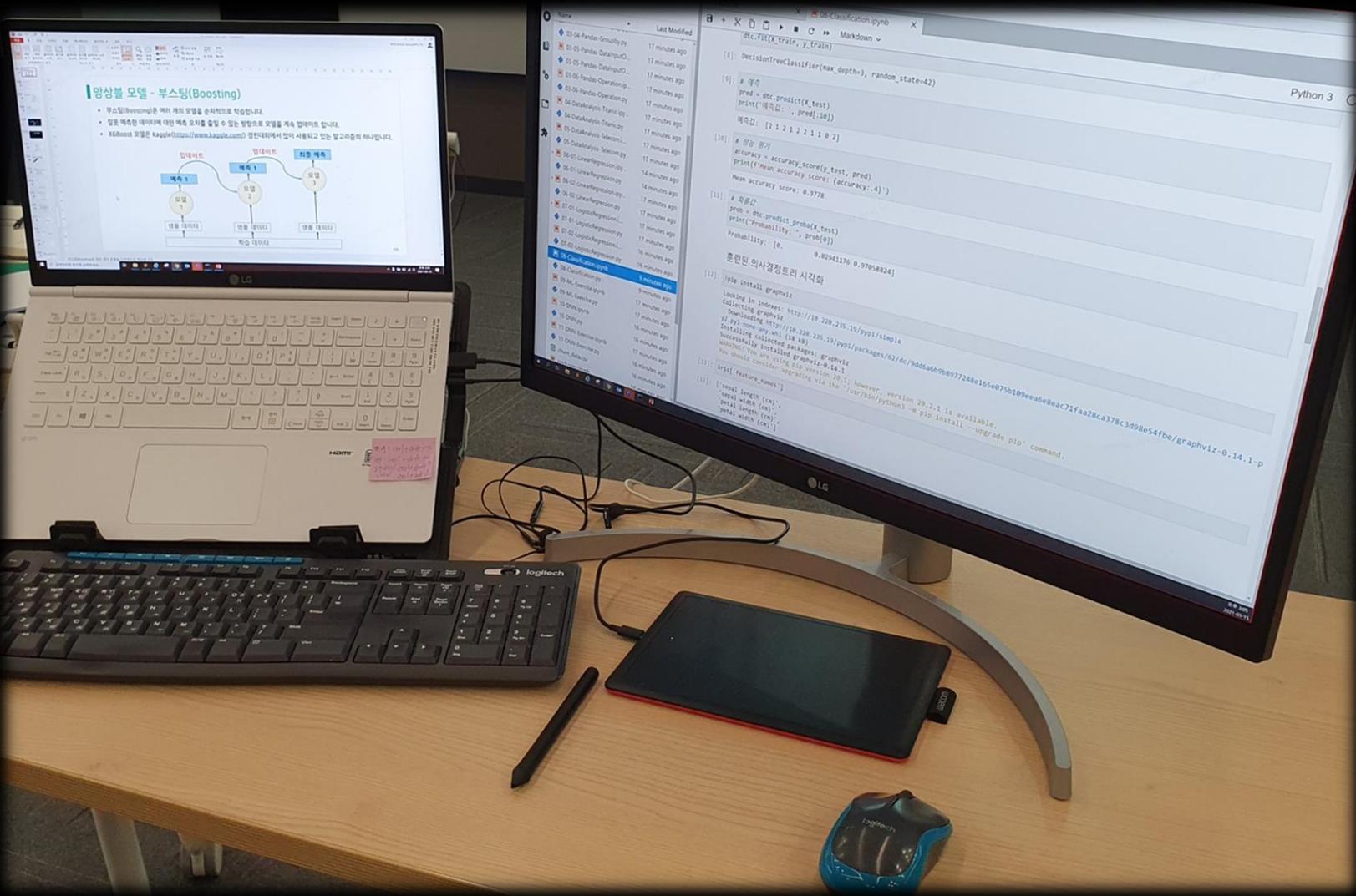
PMS(Project Management System) 개발

ADD플랫폼(AI Data Discovery Platform) 개발

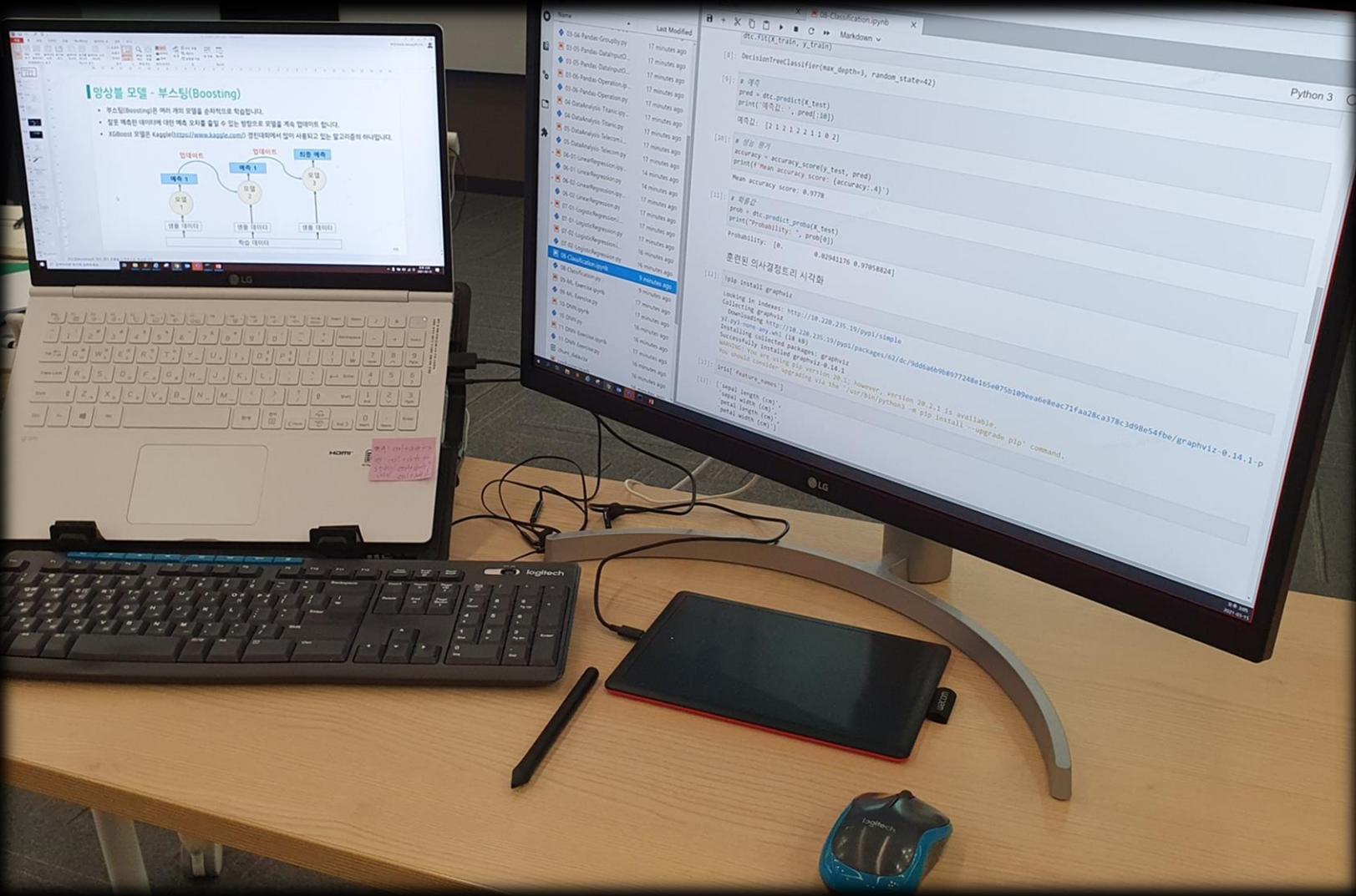
BEMS(Building Energy Management System) 개발

- 관심분야 : Self Development, Book, Overseas Travel, Investment

# 권장 실습환경



# 권장 실습환경



# 목 차

1. 코드(CODE)
2. 인공지능 개요
3. 데이터 분석과 시각화
4. 머신러닝 핵심 알고리즘
5. 스타트 딥러닝
6. Further Study

# 학습 목표

AI 분야에서 반드시 알아야 하는 핵심 개념을 단계적으로 이해합니다.

데이터를 수집하고 분석이 가능한 형태로 정리하는 기술을 습득합니다.

머신러닝 알고리즘과 딥러닝 심층신경망 원리를 파악하고 활용합니다.

인공지능 기술을 직접 실무에 활용하며 AI 전문가로 성장합니다.

# 자료

실습 파일 <https://github.com/kgpark88/ai-summary>

파이썬 스터디  
참고자료 <https://github.com/kgpark88/python>

딥러닝 스터디  
참고자료 <https://github.com/kgpark88/deeplearning>

주관식

## 다음 유튜브 신조어의 뜻을 아는 대로 쓰시오.

① 불소

⑦ 설참

② 실매

⑧ 닉차

③ 톡디

⑨ 전공

④ 반신

⑩ 임구

⑤ 반박

⑪ 깔테

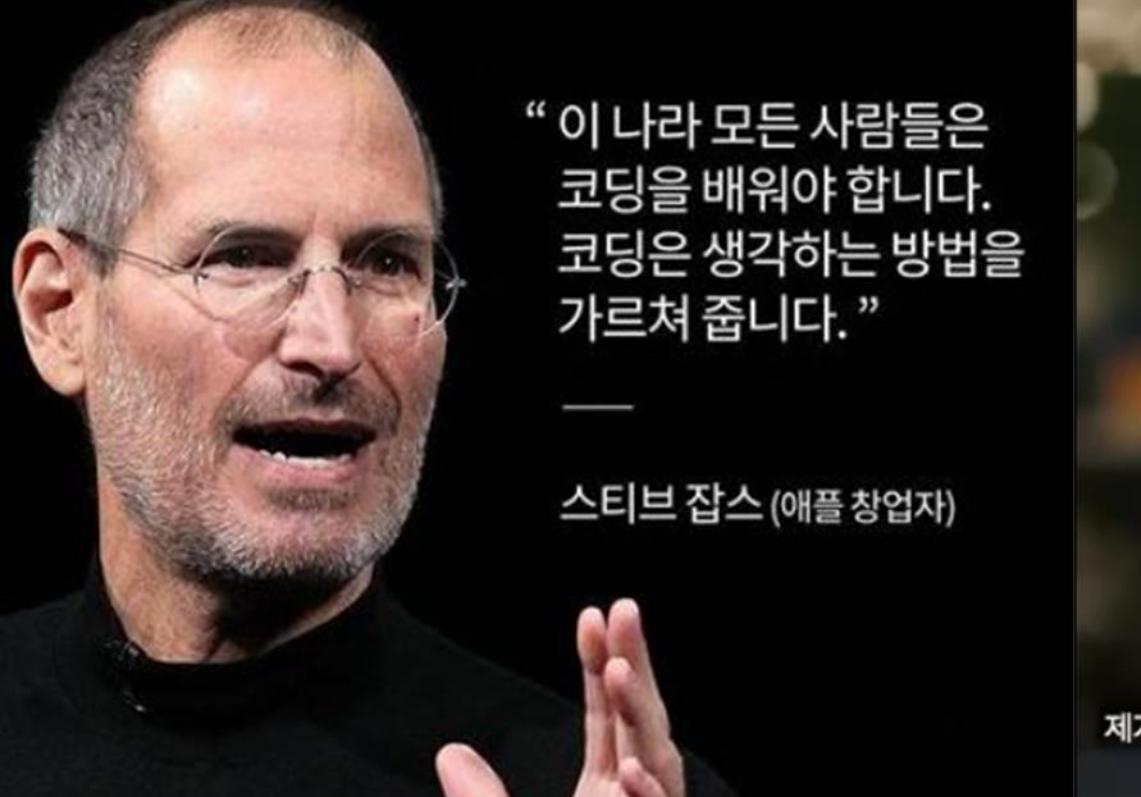
⑥ 윰차

⑫ 구취

# 1. 코드(CODE)

**컴퓨터 코드**(computer code) 또는 **프로그램 코드**(program code)는 컴퓨터가 실행하는 컴퓨터 프로그램을 구성하는 명령어들의 모임이다. 컴퓨터 하드웨어에서 실행되는 소프트웨어의 두 요소 가운데 하나이며, 다른 하나는 데이터이다.

출처 : <https://ko.wikipedia.org/>



“이 나라 모든 사람들은  
코딩을 배워야 합니다.  
코딩은 생각하는 방법을  
가르쳐 줍니다.”

스티브 잡스 (애플 창업자)



MARK  
CREATED facebook

제가 프로그래밍을 처음 배우기 시작한 이유는 컴퓨터 공학을 완습하거나 그분야의 최고가 되고 싶어서가 아니라 저를 포함한 저희집 남매들이 서로 같이 재밌게 할 수 있는 뭔가를 만들고 싶었거든요.



차세대 프로그래머는 미래의 마법사입니다. 다른 사람과 비교했을 때 마치 마법 능력이 있는 것처럼 보여질 거예요.

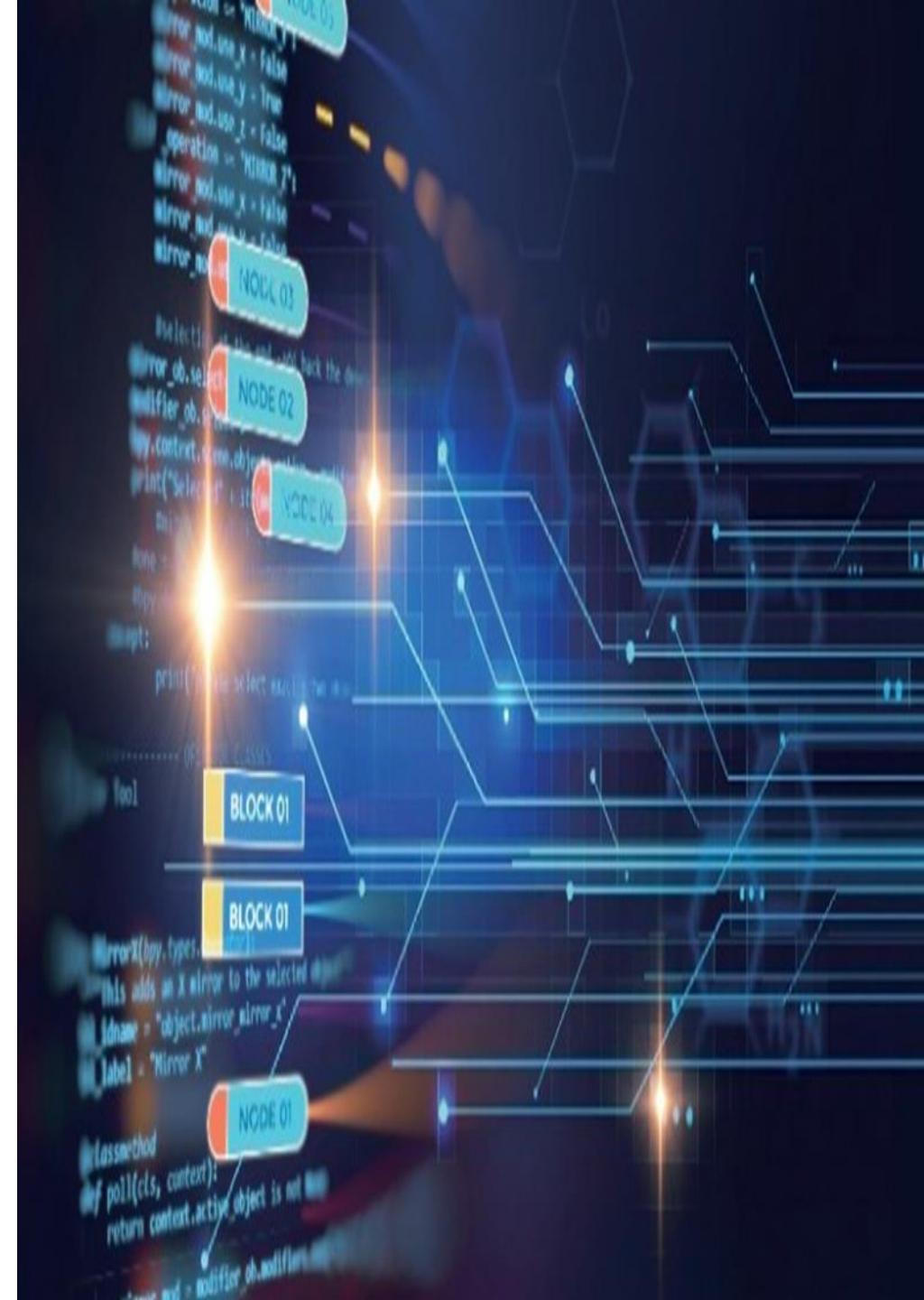


HADI  
CREATED CODE.ORG

목표가 돈을 많이 버는 것이건 세상을 바꾸는 것이건 상관없이  
컴퓨터 프로그래밍 능력은 당신에게 어마어마한 힘을 제공할 것입니다.

# 프로그래미란?

- 컴퓨터 프로그램은 특정 문제를 해결하기 위해 고안된 **특정 작업을 수행하기 위한 일련의 명령문의 집합체**
- 스마트폰, 태블릿 등에서는 ‘앱’이라는 용어를 사용
- 소프트웨어는 하드웨어의 반대 개념으로서의 의미이지만, 일반적으로는 프로그램과 같은 의미
- 프로그래밍은 주어진 문제를 해결하기 위해 컴퓨터 프로그램을 만들고 실행하는 전 과정  
절차 : 문제분석 → 입출력설계 → 알고리즘설계  
→ 코딩 → 프로그램실행



# 코딩이란?

- 코딩[Coding]은 컴퓨터 프로그램 언어로 프로그램을 작성하는 것 입니다.  
좁은 의미의 프로그래밍입니다.
- 영어를 배우는 것 만큼 코딩을 배우는 게 중요합니다.
- 프로그램 언어 규칙에 따라 글을 쓰는 것만으로 컴퓨터에게 원하는 일을 시킬 수 있습니다.
- 코딩교육을 통해 소프트웨어시대의 경쟁력을 갖출 수 있게 될 것입니다.

코딩은 문제해결을 위한  
절차적 사고와 논리적 사고력을  
키워 줍니다.



# 코딩을 배워야 하는 이유

- 소프트웨어의 발전, 특히 소프트웨어를 만드는 소프트웨어 개발툴의 발전으로  
모든 사람이 코딩을 할 수 있는 시대가 되었습니다.
- 코딩은 프로그래머 직종에 한정돼 있지 않으며,  
데이터 분석, 과학, 의학, 엔지니어링, 영업, 농작물 재배 등에도 코딩 능력이 필요합니다.
- 전 산업이 소프트웨어를 이용해서 자동화 지능화로 혁신하고 있으며,  
모든 소프트웨어는 코딩으로 만들어집니다.



You Can Create Anything You Want



Instant Scalability



Good Income

# 파이썬(Python)

The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation is a large Python logo. On the right side of the header are buttons for 'Donate', 'Search', 'GO', and 'Socialize'. The main content area has a dark blue background. On the left, there's a code editor window displaying Python code for generating a Fibonacci series up to n=1000. On the right, there's a section titled 'Functions Defined' with text about Python's function definition capabilities and a link to 'More about defining functions in Python 3'. At the bottom, there's a footer with the text 'Python is a programming language that lets you work quickly and integrate systems more effectively.' followed by a 'Learn More' link.

<https://www.python.org/>

python.org

Python PSF Docs PyPI Jobs Community

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

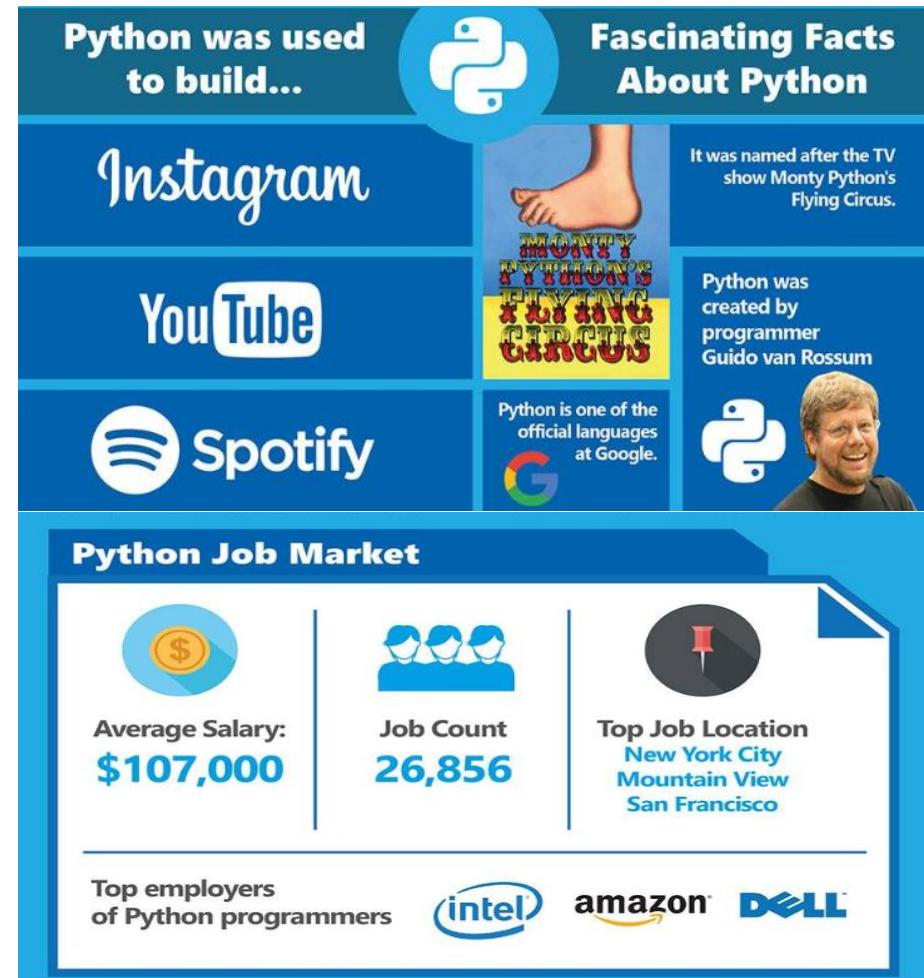
The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)

# 파이썬이 유명한 이유

- ✓ Easy to Learn and Use
- ✓ Mature and Supportive Python Community
- ✓ Support from Renowned Corporate Sponsors
- ✓ Hundreds of Python Libraries and Frameworks
- ✓ Versatility, Efficiency, Reliability, and Speed
- ✓ Big data, Machine Learning and Cloud Computing
- ✓ First-choice Language
- ✓ The Flexibility of Python Language
- ✓ Use of python in academics
- ✓ Automation



# 파이썬 주요 패키지(라이브러리)



NumPy

행렬과 다차원 배열을 쉽게 처리 할 수 있게 해주는 라이브러리

pandas

데이터를 처리하고 분석하는 데 효과적인 패키지



데이터를 차트나 플롯(Plot)으로 그려주는 시각화 패키지



matplotlib을 기반으로 다양한 색상 테마와 통계용 차트 등의 기능을 추가한 시각화 패키지



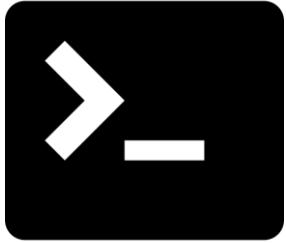
교육 및 실무를 위한 머신러닝 패키지



구글에서 만든 오픈소스 딥러닝 프레임워크

# 개발환경

## Terminal

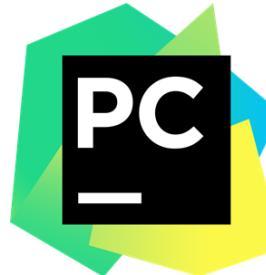


## Code Editor



Visual Studio Code

## IDE



PyCharm



Sublime Text



Spyder

## Notebook



Jupyter  
Notebook

## WEB



<https://colab.research.google.com/>



JupyterLab



<https://repl.it/>



<https://glot.io/>

# 개발환경 - Chrome

컴퓨터에 Chrome이 설치되어 있지 않은 경우, [Chrome을 다운로드하여 설치](#)하고 기본 웹브라우저를 Chrome로 설정하세요.

## 기본 웹브라우저로 Chrome 설정

Windows 10



1. 컴퓨터에서 시작 메뉴 를 클릭합니다.
2. 설정 을 클릭합니다.
3. 기본 앱을 엽니다.
  - 기존 버전: 시스템 > 기본 앱을 클릭합니다.
  - 크리에이터스 업데이트: 앱 > 기본 앱
4. 하단의 '웹 브라우저'에서 현재 브라우저를 클릭합니다. 일반적으로 Microsoft Edge입니다.
5. '앱 선택' 창에서 **Chrome**을 클릭합니다.

나중에 손쉽게 Chrome을 열려면 작업 표시줄에 단축키를 추가하세요.

1. 컴퓨터에서 Chrome을 엽니다.
2. 하단의 Windows 작업 표시줄에서 Chrome을 마우스 오른쪽 버튼으로 클릭합니다.
3. 작업 표시줄에 고정을 클릭합니다.

출처 : <https://bit.ly/30DvgKY>

# 개발환경 - 코랩(Colab)

개발툴 설치없이 구글클라우드에서 데이터분석과 AI 모델을 개발할 수 있는 환경으로 딥러닝에 필요한 GPU를 사용할 수 있습니다.

<https://colab.research.google.com>

구글 계정 필요

The screenshot shows the Google Colab interface. At the top, there's a blue header bar with the URL and an orange bar that says "구글 계정 필요". The main area has a toolbar with "Colaboratory에 오신 것을 환영합니다" and various menu items like "파일", "수정", "보기", "삽입", "런타임", "도구", "도움말". A dropdown menu is open under "런타임":

- 모두 실행 (Ctrl+F9)
- 이전 셀 실행 (Ctrl+F8)
- 초점이 맞춰진 셀 실행 (Ctrl+Enter)
- 선택항목 실행 (Ctrl+Shift+Enter)
- 이후 셀 실행 (Ctrl+F10)
- 실행 중단 (Ctrl+M I)
- 런타임 다시 시작 (Ctrl+M R)
- 다시 시작 및 모두 실행
- 런타임 초기화
- 런타임 유형 변경
- 세션 관리
- 런타임 로그 보기

A modal window titled "노트 설정" (Notebook Settings) is overlaid on the interface. It contains the following text:

Colaboratory의 주요 기능을 간단하게 알아보세요

하드웨어 가속기  
GPU ?

Colab를 최대한 활용하려면 필요하지 않은 경우 GPU를 사용하지 않는 것이 좋습니다.  
[자세히 알아보기](#)

이 노트를 저장할 때 코드 셀 출력 생략

취소      저장

At the bottom left, there's a "Coding TensorFlow" logo with a yellow 'F' icon.



고성능GPU(Graphics Processing Unit)

# 파이썬 기초

## ■ 변수 할당(Variable Assignment)

```
x = 2
```

```
y = 3
```

```
z = x + y
```

```
x = 'hello'
```

```
x = "hello"
```

```
x
```

```
[Out] 'hello'
```

Single Quotation  
작은 따옴표

Double Quotation  
쌍 따옴표

## ■ 출력

```
print(x)
```

```
[Out] 'hello'
```

## ■ 리스트(List)

```
[1, 2, 3]
```

```
['a', 'b', 'c']
```

```
my_list = [1, 2, 'apple', True]
```

```
my_list.append(100)
```

```
my_list[0]
```

```
my_list[:-1]
```

```
my_list[-1]
```

Bracket  
대괄호

## ■ 딕셔너리(Dictionary)

```
d = {'key1': 'item1', 'key2': 'item2'}
```

```
d['key1']
```

```
[Out] 'item1'
```

Brace  
중괄호

# 파이썬 실습



[https://github.com/kgpark88/ai-summary/blob/main/01\\_Python.ipynb](https://github.com/kgpark88/ai-summary/blob/main/01_Python.ipynb)

kgpark88 / ai-summary

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main / ai-summary / 01\_Python.ipynb Go to file ...

kgpark88 Colaboratory를 통해 생성됨 Latest commit 59fc46e 6 minutes ago History

1 contributor

2413 lines (2413 sloc) | 49.5 KB

Open in Colab

## 파이썬 기본

- 데이터 타입
  - 숫자(Number)
  - 문자열(String)

## 2. AI 개요

인공지능 또는 AI는 인간의 학습능력, 추론능력, 지각능력,  
그 외에 인공적으로 구현한 컴퓨터 프로그램 또는 이를 포함한 컴퓨터 시스템이다.

출처 : <https://ko.wikipedia.org/>

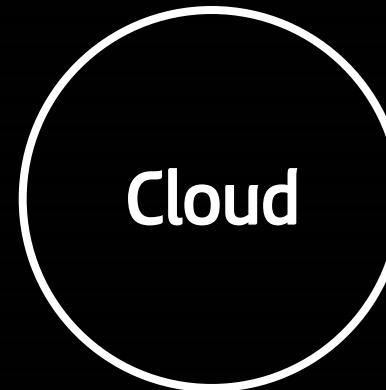
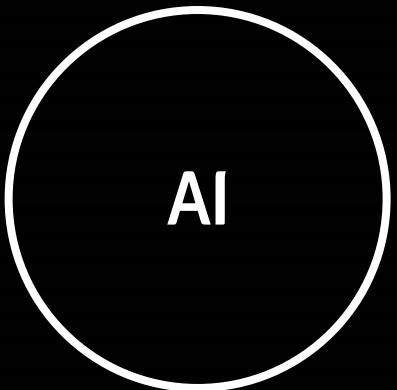
# Tech and Change

증기기관  
내연기관

전기  
에너지

컴퓨터  
인터넷

# Tech and Change



iamai





# 인공지능 활용사례 - 이미지 분류

이미지넷(ImageNet) 제공 이미지 데이터  
1,000여 카테고리로 분류된 100만 개의 이미지

airplane



automobile



bird



cat



deer



dog



frog



horse



ship

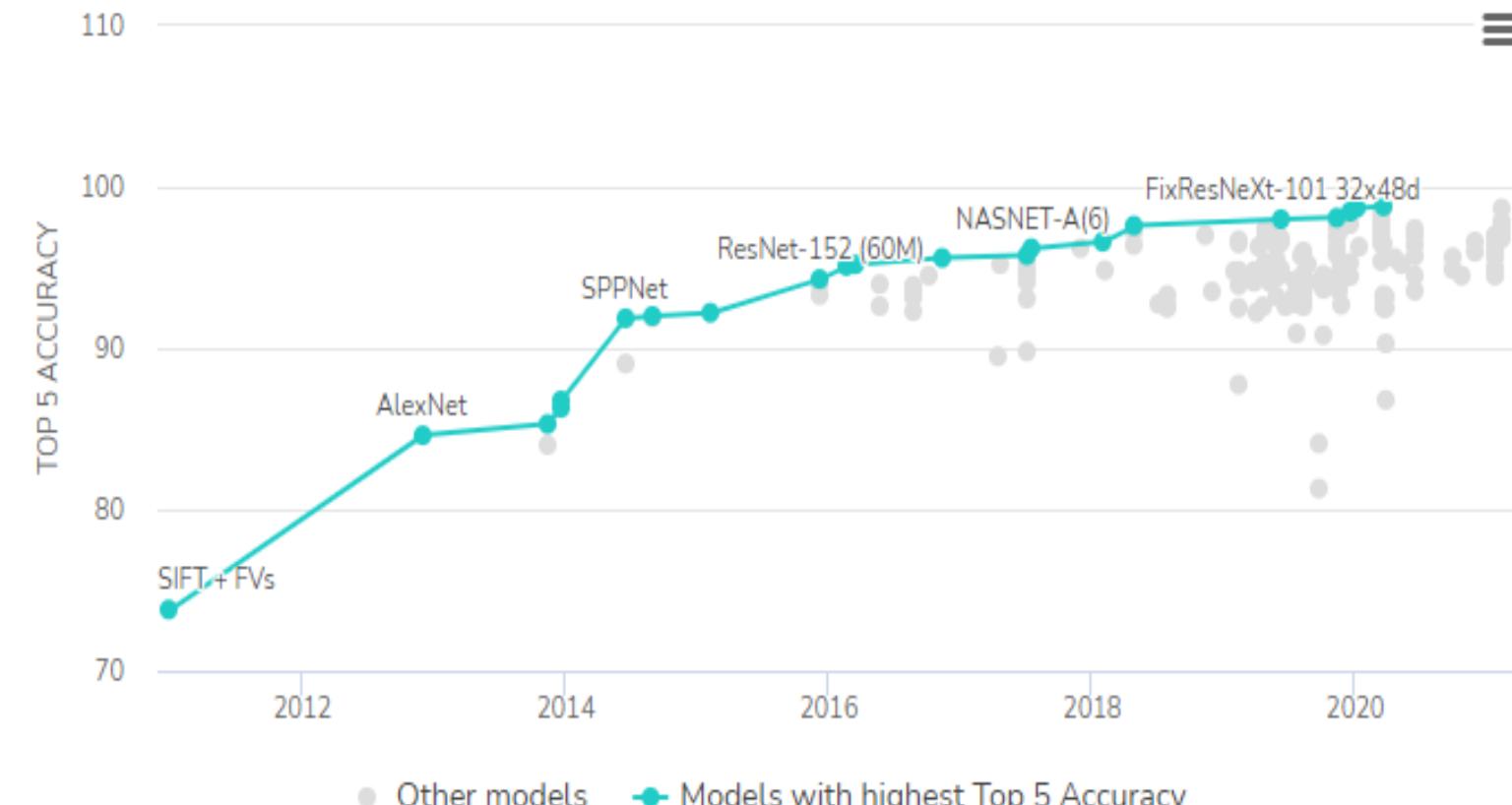


truck

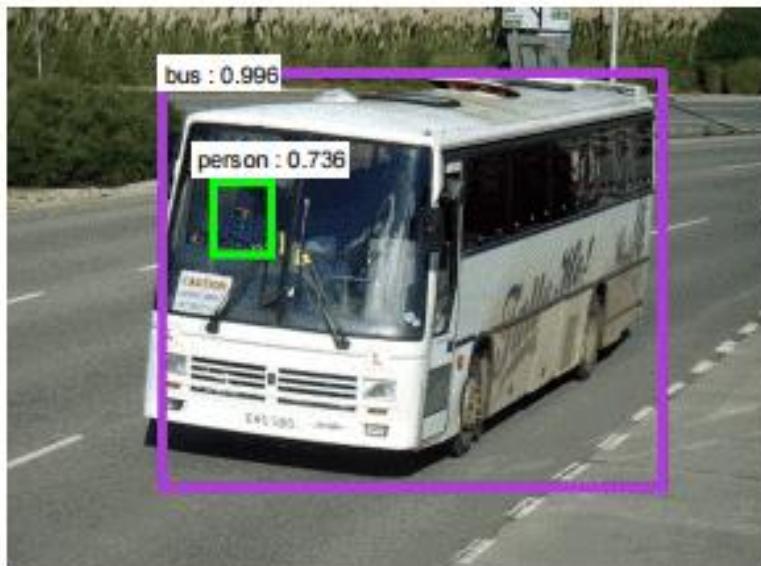
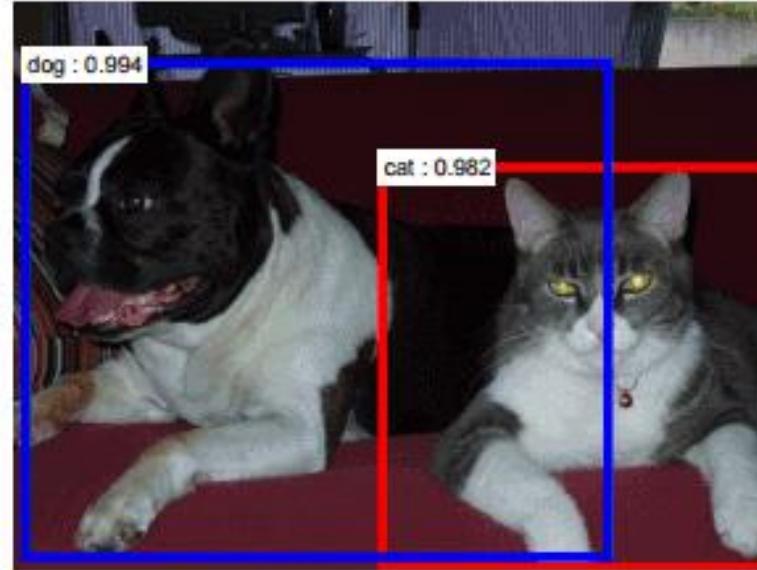
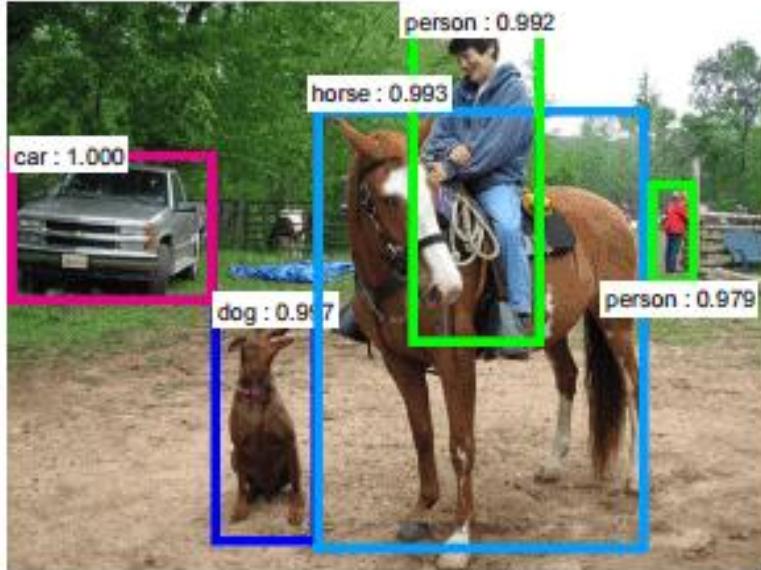


Leaderboard

Dataset



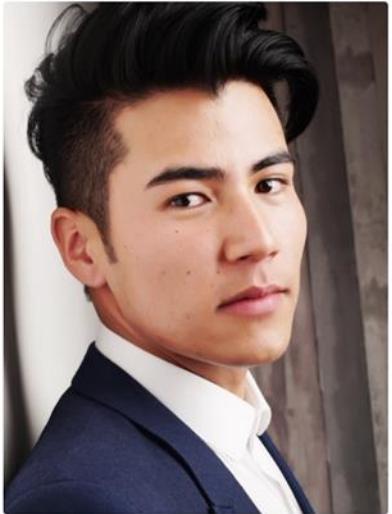
# 인공지능 적용사례 - 객체 탐지(Object Detection)



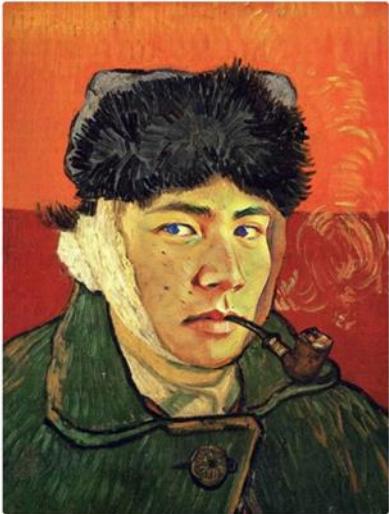
출처 : <https://sigmoidal.io/dl-computer-vision-beyond-classification>

# 인공지능 활용사례 - 이미지 생성(Style Transfer)

ORIGINAL PHOTO



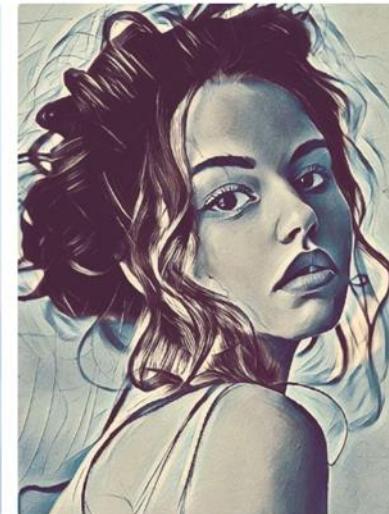
REWORKED PHOTO



ORIGINAL PHOTO



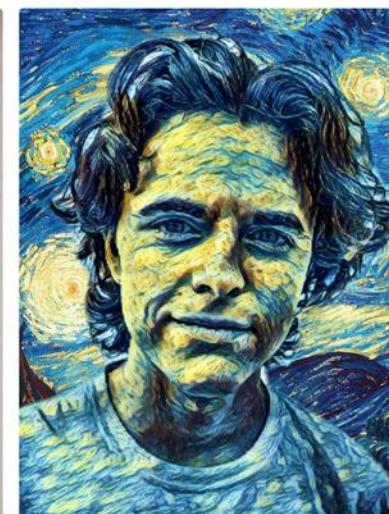
REWORKED PHOTO



ORIGINAL PHOTO



REWORKED PHOTO



ORIGINAL PHOTO



REWORKED PHOTO



ORIGINAL PHOTO



REWORKED PHOTO



ORIGINAL PHOTO



REWORKED PHOTO



# 인공지능 활용사례 - 이미지 생성(GAN: generative adversarial network)



Original



Change Hair Color



Change Eye Color



Change Hair Style

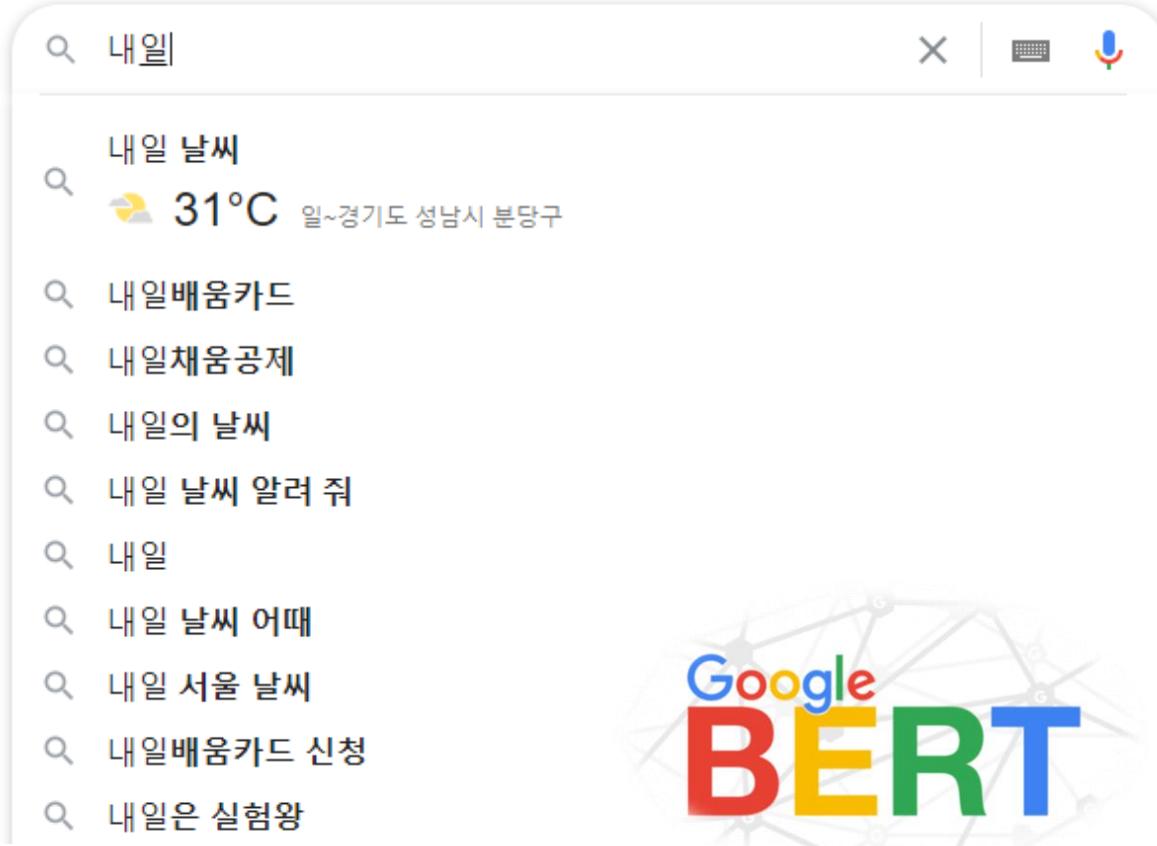


Open Mouth



Add Assesories

# 인공지능 활용사례 - 자연어 처리



# 인공지능 활용사례 - Improving our world with AI



# 인공지능(Artificial Intelligent)



## 인공 지능

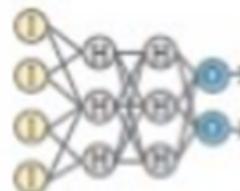
인간의 지적능력(추론, 인지)을 구현하는 모든 기술



## 머신 러닝

알고리즘으로 데이터를 분석, 학습하여 판단이나 예측을 하는 기술

선형회귀  
로지스틱회귀  
K-최근접 이웃  
결정트리  
랜덤포레스트  
서포트 벡터 머신



클러스터링 차원축소

## 딥러닝

인공신경망 알고리즘을 활용하는 머신러닝 기술

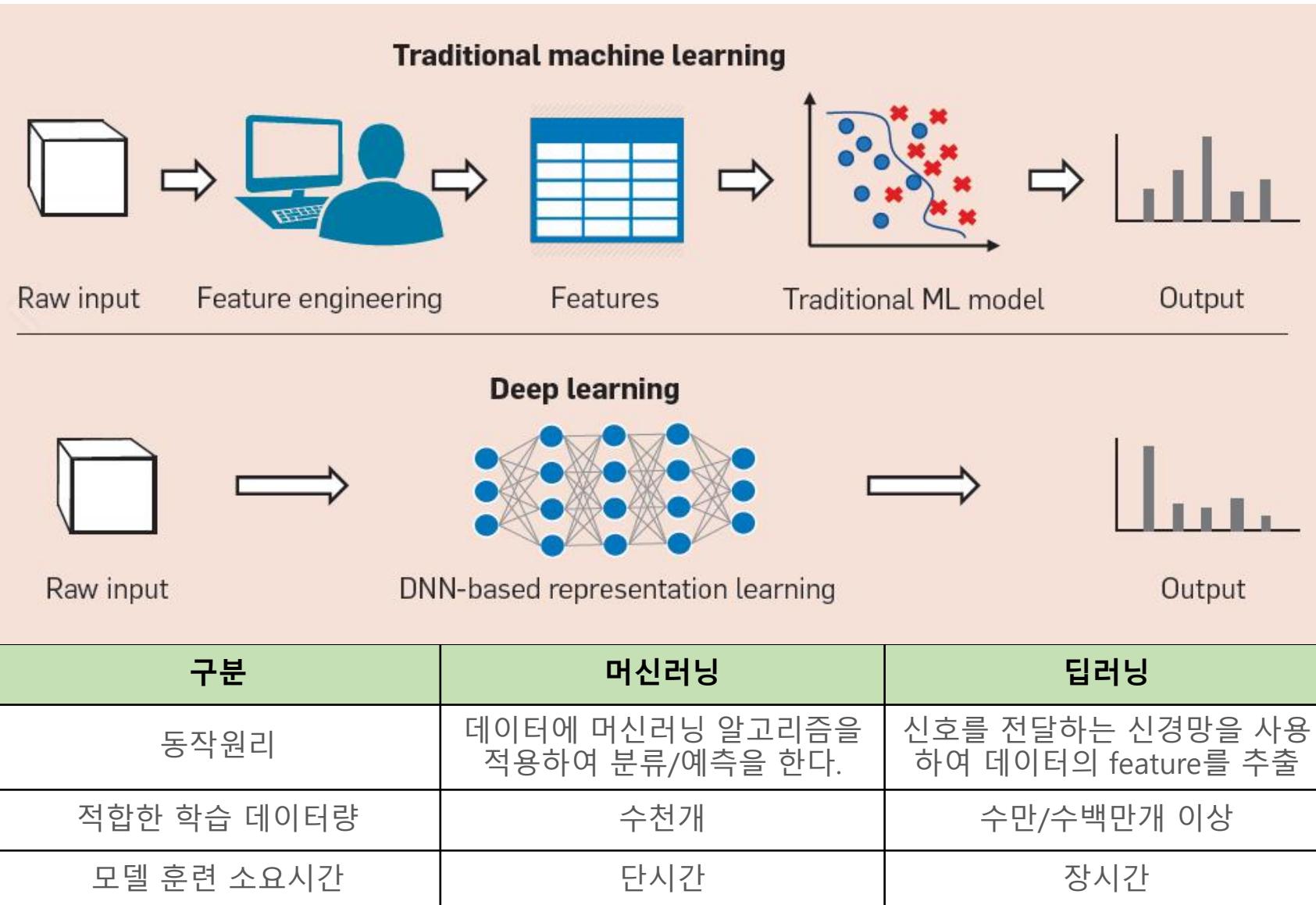
심층신경망  
(DNN)

합성곱 신경망  
(CNN)

순환 신경망  
(RNN)

강화 학습

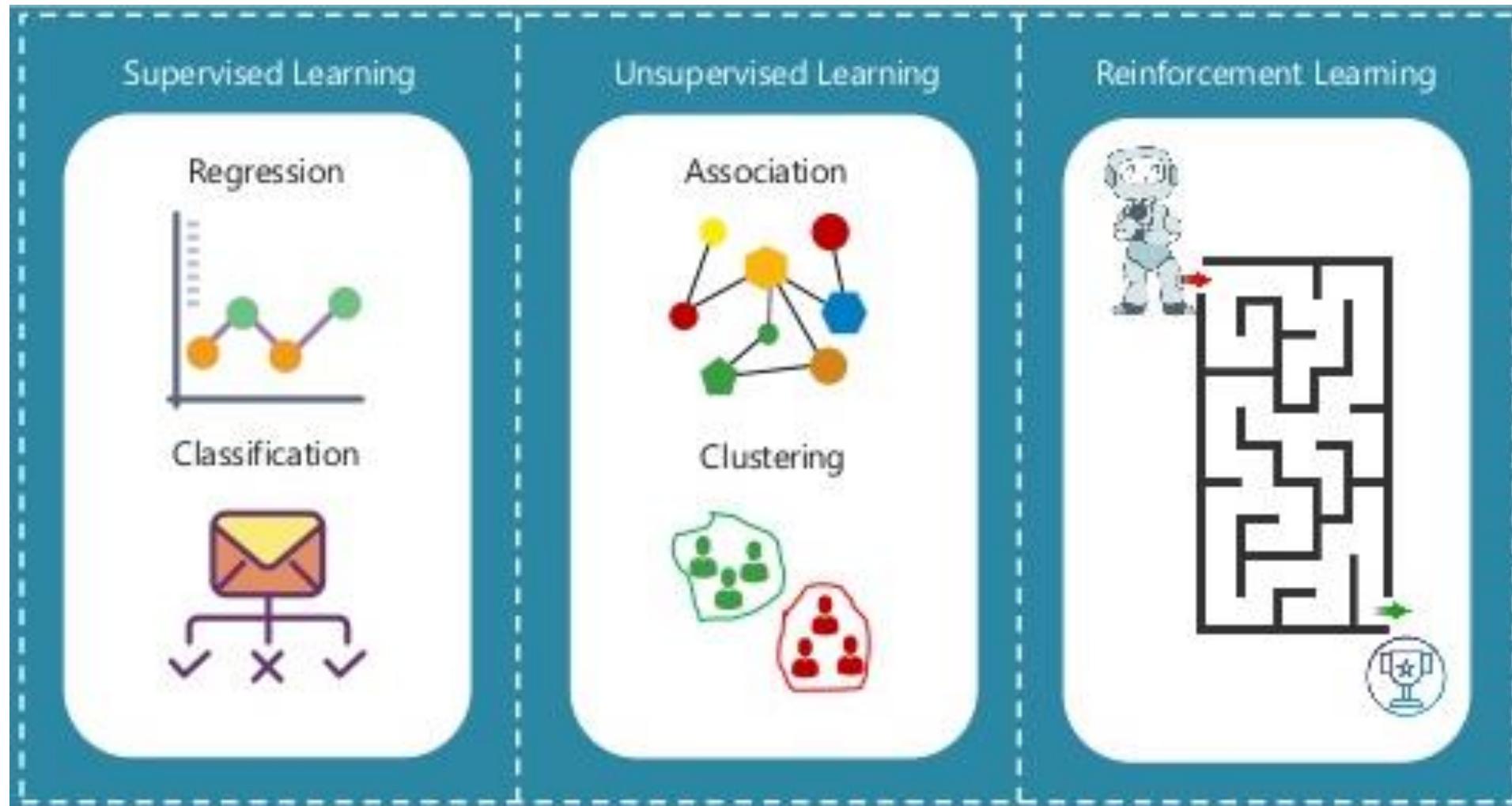
# 머신러닝 VS 딥러닝



머신러닝에서는 데이터로부터 속성(Feature)을 찾아내는 역할을 컴퓨터(Machine)가 담당

딥러닝에서는 신경망으로 데이터/이미지를 ‘있는 그대로’ 학습하며, 데이터에 포함된 중요한 속성을 컴퓨터가 스스로 학습

# 머신러닝/딥러닝 학습 방법



정답지(Label)로 학습  
분류(Classification)  
예측(Regression)

정답지(Label) 없이 학습  
군집(Clustering)  
차원 축소

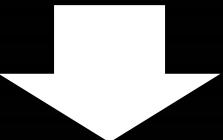
시뮬레이션 반복 학습  
성능 강화 등에 사용  
마르코프 결정 과정(Markov Decision Process)

# AI 시대의 경쟁력

문제의 본질을 파악하는 능력과 데이터를 만드는 능력이 중요

인공지능을 활용하여 기존의 나의 일을 효율화 하는 것이 실력

AI를 활용하여 기존의 일을 효율적으로 바꾸는 일을 주도하는 것이 경쟁력



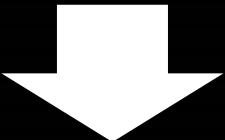
*Let's start my game!*

# 3. 데이터 분석과 시각화

**빠르게 쌓여가는 방대한 데이터로 빅데이터(Big Data) 시대**

**빅데이터 저장, 처리, 분석에 필요한 컴퓨팅 자원을 저렴한 비용으로 사용 가능**

**컴퓨팅 파워의 대중화는 최적의 학습 환경과 연구 인프라를 제공**



**데이터 자체가 가장 중요한 자원이며, 데이터를 수집하고 분석이 가능한 형태로 정리하는 것이 중요**

**관점의 변화가 필요하다.**

# 데이터 활용

데이터 분석을 통한 인사이트 발굴 및 의사결정에 활용



기존 데이터를 사용해서 새로운 가치가  
더해진 데이터(Value Added DATA)를 만들어 낸다. ('예측 확률 값')

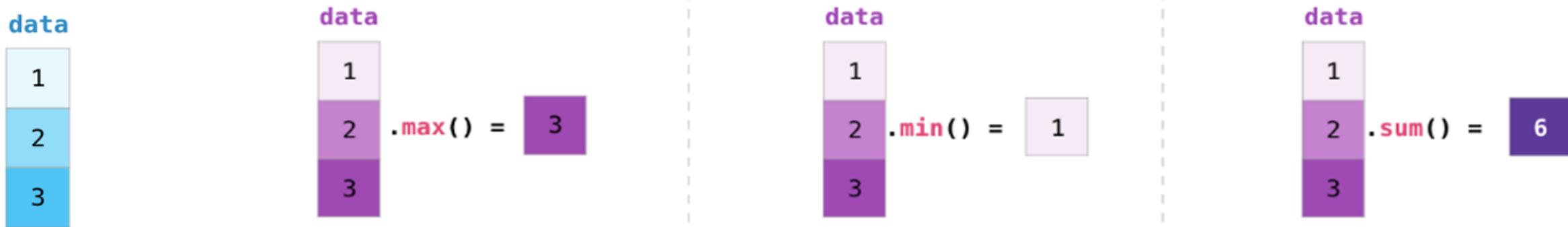


새로운 가치 데이터를 바탕으로 비즈니스적으로 의미 있는 변화를 만든다.  
기존 업무와 프로세스를 변경하여 매출 증대, 비용 절감, 효율 증대 등의 변화를 만든다.

# 넘파이(Numpy)

NumPy(Numerical Python)는 데이터 분석, 수학/과학연산을 위한 파이썬 기본 패키지로 고성능의 다차원 배열 객체와 다양한 객체에 대해 고속 연산을 가능하게 합니다.

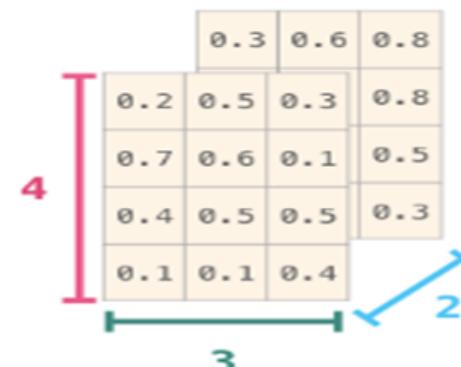
```
data = np.array([1,2,3])
```



```
np.array([[1,2],[3,4],  
         [[5,6],[7,8]]])
```

1	2	5
3	4	6
		8

```
np.random.random((4,3,2))
```



# 넘파이(Numpy)



[https://github.com/kgpark88/ai-summary/blob/main/02\\_Numpy.ipynb](https://github.com/kgpark88/ai-summary/blob/main/02_Numpy.ipynb)

## ■ Numpy 라이브러리 임포트

```
import numpy as np
```

## ■ Numpy Array 생성

```
my_list = [1, 2, 3]
np.array(my_list)
[Out] array([1, 2, 3])
```

```
my_matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
np.array(my_matrix)
[Out] array([[1, 2, 3],
             [4, 5, 6],
             [7, 8, 9]])
```

# 데이터 분석 필수 라이브러리 판다스(Pandas)

판다스(Pandas)는 데이터 처리와 분석을 위해 널리 사용되는 파이썬 라이브러리

데이터사이언티스트에게 필요한 기본적이면서도 아주 중요한 도구

행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있음

데이터를 수집하고 정리하는 데 최적화 된 도구



판다스는 시리즈(Series)와 데이터프레임(DataFrame)이라는 구조화된 데이터 형식을 제공

시리즈(Series) : 1차원 배열

데이터프레임(DataFrame) : 2차원 배열

# 판다스 시리즈(Series)

1차원의 배열의 값(values)과 각 값에 대응하는 인덱스(index)를 부여할 수 있는 데이터 구조

```
import pandas as pd
```

```
sr = pd.Series([20000, 18000, 5000])  
print(sr)
```

index	values
피자	20000
치킨	18000
맥주	5000
dtype: int64	

```
sr = pd.Series([20000, 18000, 5000], index = ['피자', '치킨', '맥주'])  
print(sr.index)  
print(sr.values)  
print(sr)
```

```
sr = pd.Series({'피자': 20000, '치킨': 18000, '맥주': 5000})  
print(sr)
```

# 판다스 데이터프레임(DataFrame)

데이터프레임은 행과 열을 가지는 자료구조로 인덱스(index), 열(columns), 값(values)으로 구성

열(columns)

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
인덱스 (index)									
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic

값(values)

# 판다스 실습



<https://github.com/kgpark88/ai-summary>

- [03\\_01\\_Pandas\\_Series.ipynb](#)
- [03\\_02\\_Pandas\\_DataFrame.ipynb](#)
- [03\\_03\\_Pandas\\_MissingData.ipynb](#)
- [03\\_04\\_Pandas\\_Groupby.ipynb](#)
- [03\\_05\\_Pandas\\_DataInputOutput.ipynb](#)
- [03\\_06\\_Pandas\\_Operation.ipynb](#)

# 판다스 Exercise

<https://bit.ly/3bnwEHT>





**Question 1 – Define Python Pandas.**

**Question 2 – What Are The Different Types Of Data Structures In Pandas?**

**Question 6 – What Are The Most Important Features Of The Pandas Library?**

**Question 8 – What are the different ways of creating DataFrame in pandas?  
Explain with examples.**

**Question 9 – Explain Categorical Data In Pandas?**

**Question 14 – How Can You Iterate Over Dataframe In Pandas?**

**Question 17 – What Is Groupby Function In Pandas?**

# 데이터 분석 실습 - 타이타닉 데이터셋



[https://github.com/kgpark88/ai-summary/blob/main/04\\_DataAnalysis\\_Titanic.ipynb](https://github.com/kgpark88/ai-summary/blob/main/04_DataAnalysis_Titanic.ipynb)

## ■ Seaborn 라이브러리 임포트

```
import seaborn as sns
```

## ■ 파일에서 데이터를 로드

```
df = sns.load_dataset('titanic')
```

## ■ 데이터 확인

```
df.head()
```

```
df.head(20)
```

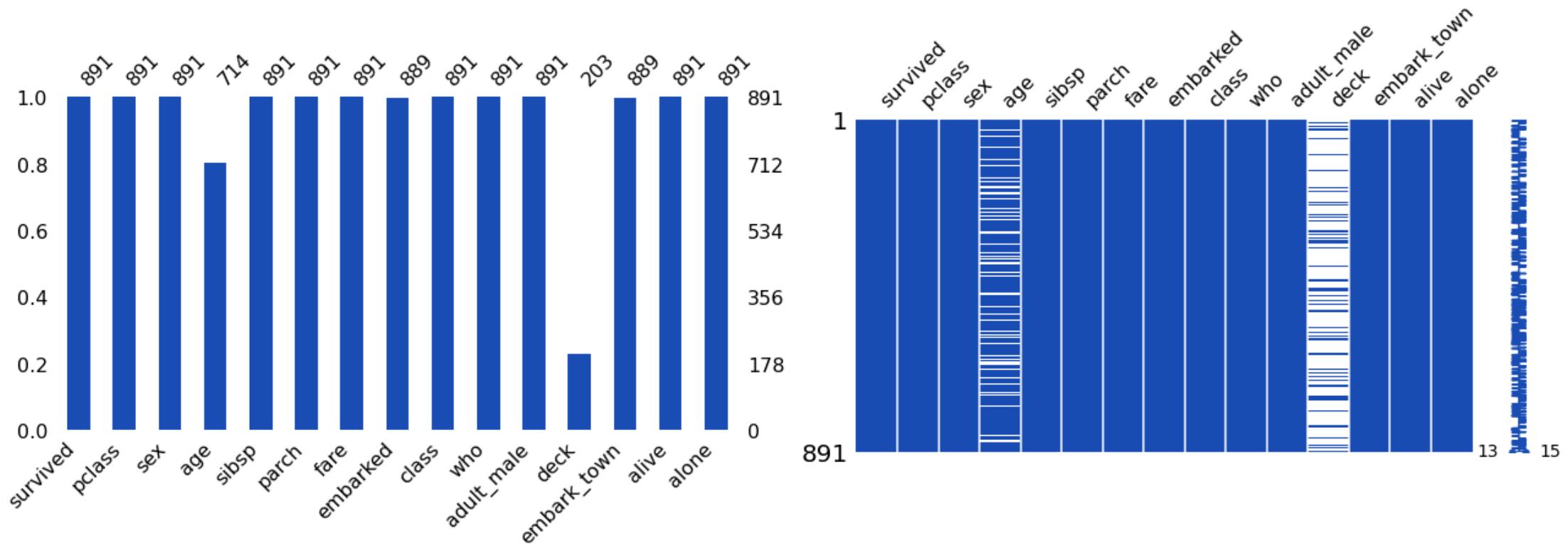
```
df.tail()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

# 데이터 분석 실습 - 타이타닉 데이터셋

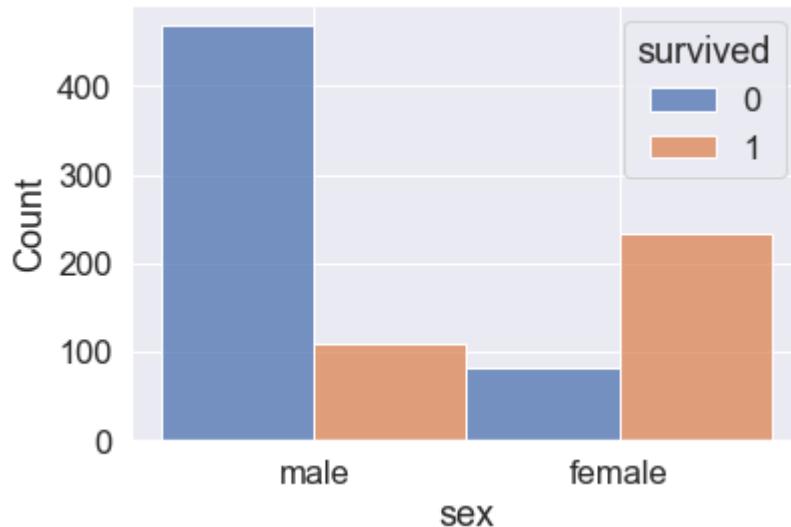
## ■ 결측값 확인

```
!pip install missingno  
import missingno as msno  
msno.bar(df, figsize=(10, 5), color=(0.1, 0.3, 0.7))  
msno.matrix(df, figsize=(10, 5), color=(0.1, 0.3, 0.7))
```

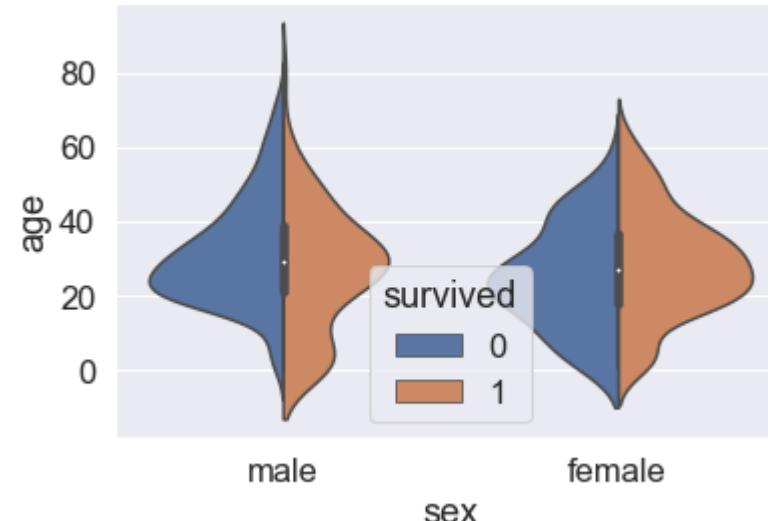


# 데이터 분석 실습 - 타이타닉 데이터셋

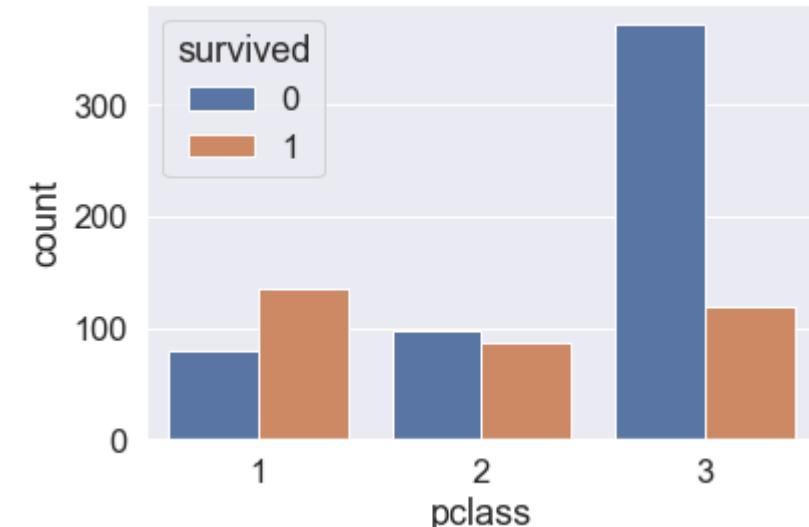
■ 성별(sex)에 따른 생존율 분포



■ 승객 나이와 생존 여부와의 관계



■ 객실등급과 생존율



```
sns.histplot(x='sex', hue='survived', multiple='dodge', data=df)
```

```
sns.violinplot(x='sex', y='age', hue='survived', data=df, split=True)
```

```
sns.countplot(x='pclass', hue='survived', data=df)
```

# 데이터 분석 실습 - 통신사 이탈고객 데이터셋



[https://github.com/kgpark88/ai-summary/blob/main/05\\_DataAnalysis\\_Telecom.ipynb](https://github.com/kgpark88/ai-summary/blob/main/05_DataAnalysis_Telecom.ipynb)

## ■ Pandas 라이브러리 임포트

```
import pandas as pd
```

## ■ 파일에서 데이터 로드

```
df = pd.read_csv('churn_data.csv')
```

## ■ 데이터 확인

```
df.head()
```

```
df.head(20)
```

```
df.tail()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No

# 데이터 분석 실습 - 통신사 이탈고객 데이터셋

## ■ 데이터구조 파악

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   customerID      7043 non-null    object  
 1   gender           7043 non-null    object  
 2   SeniorCitizen   7043 non-null    int64  
 3   Partner          7043 non-null    object  
 4   Dependents       7043 non-null    object  
 5   tenure           7043 non-null    int64  
 ....., 
 18  MonthlyCharges  7043 non-null    float64
 19  TotalCharges    7043 non-null    object  
 20  Churn            7043 non-null    object  
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

## ■ 데이터 타입 확인

`df.dtypes`

```
customerID        object
gender            object
SeniorCitizen    int64
Partner           object
Dependents        object
tenure            int64
PhoneService      object
MultipleLines     object
InternetService   object
OnlineSecurity    object
OnlineBackup      object
DeviceProtection  object
TechSupport       object
StreamingTV       object
StreamingMovies   object
Contract          object
PaperlessBilling  object
PaymentMethod     object
MonthlyCharges    float64
TotalCharges      object
Churn             object
dtype: object
```

## ■ Null 데이터 확인

`df.isnull().sum()`

```
customerID      0
gender          0
SeniorCitizen   0
Partner          0
Dependents       0
tenure           0
PhoneService     0
MultipleLines    0
InternetService  0
OnlineSecurity   0
OnlineBackup     0
DeviceProtection 0
TechSupport      0
StreamingTV      0
StreamingMovies  0
Contract          0
PaperlessBilling 0
PaymentMethod    0
MonthlyCharges   0
TotalCharges     0
Churn             0
dtype: int64
```

# 데이터 분석 실습 - 통신사 이탈고객 데이터셋

## ■ 통계 정보

df.describe()

	SeniorCitizen	tenure	MonthlyCharges
<b>count</b>	7043.000000	7043.000000	7043.000000
<b>mean</b>	0.162147	32.371149	64.761692
<b>std</b>	0.368612	24.559481	30.090047
<b>min</b>	0.000000	0.000000	18.250000
<b>25%</b>	0.000000	9.000000	35.500000
<b>50%</b>	0.000000	29.000000	70.350000
<b>75%</b>	0.000000	55.000000	89.850000
<b>max</b>	1.000000	72.000000	118.750000

## ■ 데이터 상관관계 분석

df.corr()

	SeniorCitizen	tenure	MonthlyCharges
SeniorCitizen	1.000000	0.016567	0.220173
tenure	0.016567	1.000000	0.247900
MonthlyCharges	0.220173	0.247900	1.000000

# 데이터 분석 /전처리 실습 - 통신사 이탈고객 데이터셋

## ■ 데이터 전처리

입력 데이터에서 제외: drop()

Null 데이터 처리: dropna(), fillna()

누락데이터 처리: replace()

데이터타입 변환 : astype()

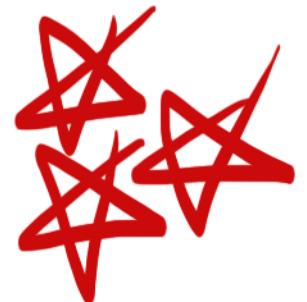
특성 추출 (feature engineering) : df[ 'new\_feature' ] = df[ 'f\_1' ]/df[ 'f\_2' ]

```
df.drop('customerID', axis=1, inplace=True)
```

```
df['TotalCharges'].replace([' '], [0], inplace=True)
```

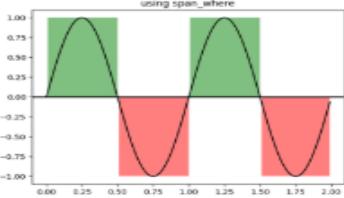
```
df['TotalCharges'] = df['TotalCharges'].astype(float)
```

```
df['Churn'].replace(['Yes', 'No'], [1, 0], inplace=True)
```

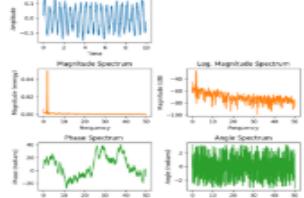


# 데이터 시각화 - 맷플롯립(Matplotlib)

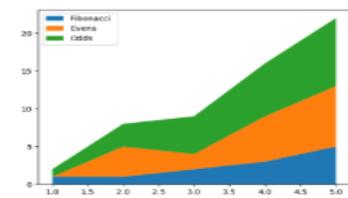
데이터를 차트나 플롯(Plot)으로 표시할 때 가장 많이 사용되는 데이터 시각화 라이브러리



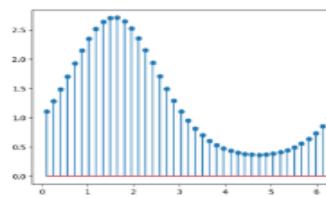
Using span\_where



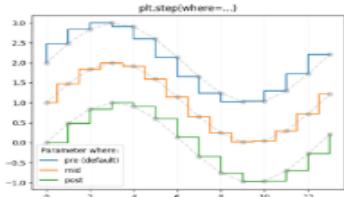
Spectrum Representations



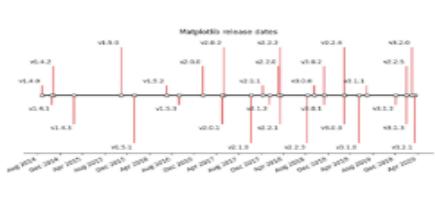
Stackplot Demo



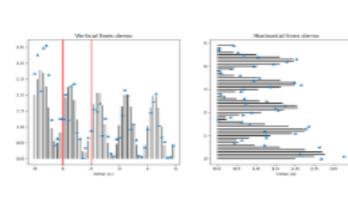
Stem Plot



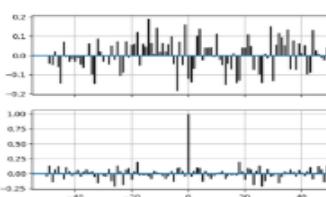
Step Demo



Creating a timeline with lines, dates, and text



hlines and vlines



Cross- and Auto-Correlation Demo

# 데이터 시각화 - 맷플롯립(Matplotlib)

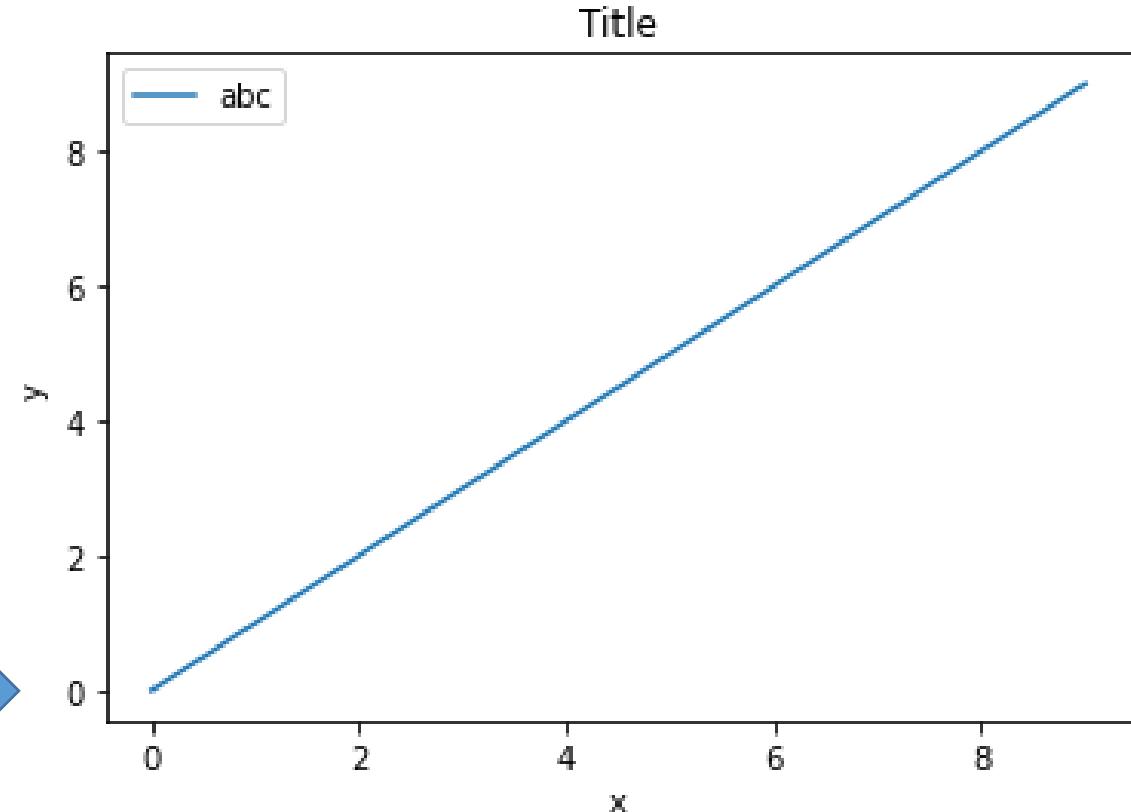
## ■ 라이브러리 임포트

```
import matplotlib.pyplot as plt  
%matplotlib inline
```

## ■ Matplotlib 사용법(예시)

```
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

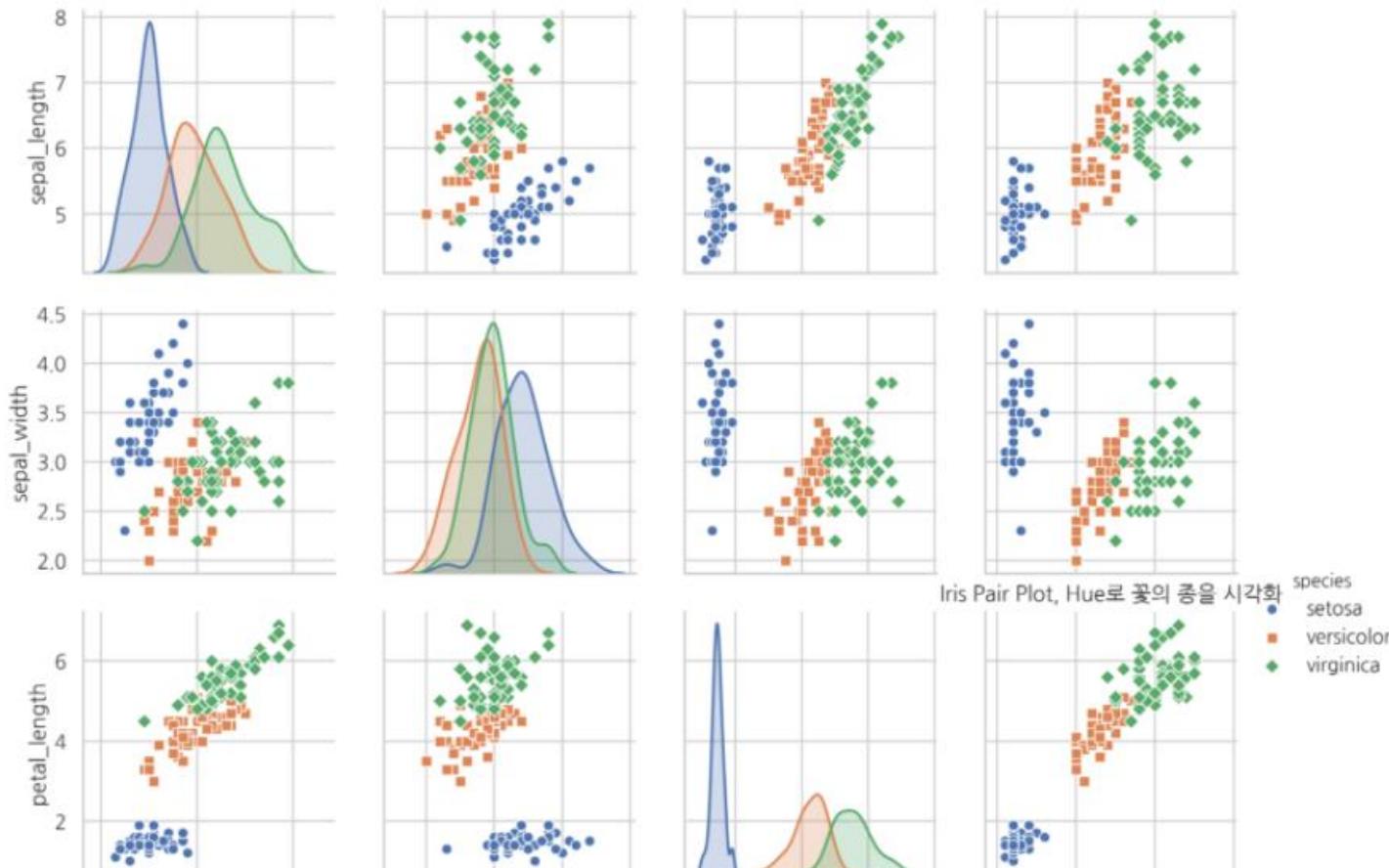
```
plt.plot(x, y)  
plt.title('Title')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.legend(['abc'])  
plt.show()
```



# 데이터 시각화 - 씨본(Seaborn)

Matplotlib을 기반으로 다양한 색상 테마와 통계용 차트 등의 기능을 추가한 시각화 라이브러리

```
sns.pairplot(iris, hue="species", markers=["o", "s", "D"])
plt.title("Iris Pair Plot, Hue로 꽃의 종을 시각화")
plt.show()
```



출처 : <https://bit.ly/3mqQsNY>

# 데이터 시각화 - 씨본(Seaborn)

## ■ 패키지 설치

```
!pip install seaborn
```

## ■ 라이브러리 임포트

```
import seaborn as sns
```

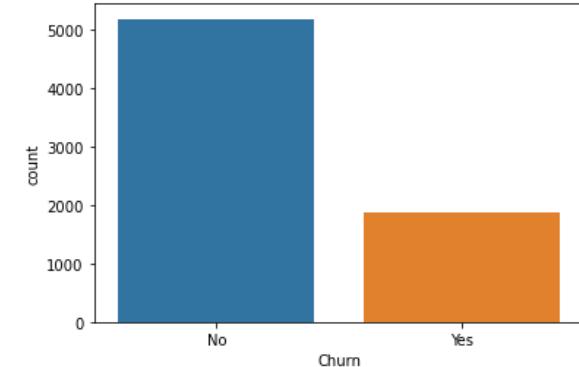
## ■ 상관관계 히트맵

```
sns.heatmap(df.corr(), annot=True)
```



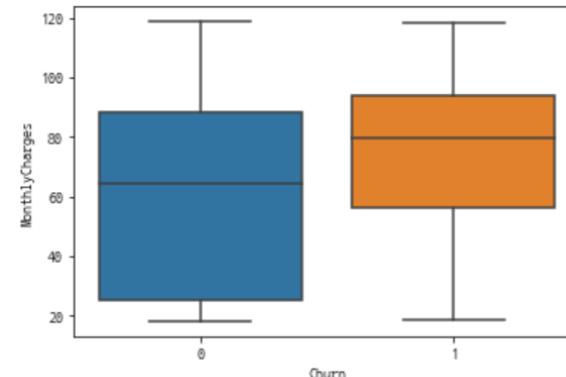
## ■ 카운트 플롯

```
sns.countplot(x='Churn', data=df)
```



## ■ 박스 플롯

```
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
```



통계적 사고관을 갖추고  
데이터 과학 기초를 이해하면  
변화하는 세상을 살아가는데  
분명 도움이 될 겁니다.



데이터 과학자 권재명

# 4. 머신러닝 핵심 알고리즘

# 데이터

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4



# 머신러닝

머신러닝은 컴퓨터 알고리즘이 데이터를 학습하여 입력과 출력간의 관계를 찾는 과정입니다.  
학습할 때 정답 레이블을 있는지 여부에 따라 지도학습과 비지도학습으로 구분을 합니다.

## ■ 지도학습(supervised learning)

- 학습시 정답을 알려 주면서 진행하는 학습으로, 학습시 데이터와 레이블(정답)이 함께 제공됩니다.
- **레이블(Label)** = 정답, 실제값, 타깃, 클래스,  $y$
- 예측된 값 = 예측값, 분류값,  $\hat{y}$  ( $y$  hat)
- 데이터마다 레이블을 달기 위해 많은 시간을 투자해야 합니다.
- 지도학습 **모델**에는 **분류모델**(이진분류, 다중분류)과 **회귀모델**(주가예측 등)이 있습니다.

## ■ 비지도학습(unsupervised learning)

- 레이블(정답) 없이 진행되는 학습으로, 데이터 자체에서 패턴을 찾아내야 할 때 사용합니다.
- 비지도학습의 대표적인 예는 군집화(clustering)와 차원축소가 있습니다.

# 지도학습

지도 학습은 정답이 있는 데이터를 활용해 데이터를 학습시키는 것입니다. 입력 값(X data)이 주어지면 입력값에 대한 Label(Y data)를 주어 학습시키며 분류모델과 회귀모델이 있습니다.

## ■ 모델

- 데이터들의 패턴을 대표할 수 있는 함수, 예)  $f(x) = ax + b$
- 함수의 입력은 독립변수이고 출력은 종속변수로, 독립변수들에 의해 출력값이 정해집니다.

## ■ 분류 모델(Classification)

- 레이블의 값들이 이산적으로 나눠질 수 있는 문제에 사용합니다.
- 예) 스팸 메일 분류, 품종 분류

Classification



## ■ 회귀 모델(Regression)

- 레이블의 값들이 연속적인 문제에 사용합니다.
- 예) 날씨 예측, 주가 예측

Regression



# 지도학습 데이터셋 구조

각 열(column)을 특징/속성(feature) 이라고 합니다.  
데이터 컬럼(column)중에 하나를 선택해서 레이블로 사용합니다.

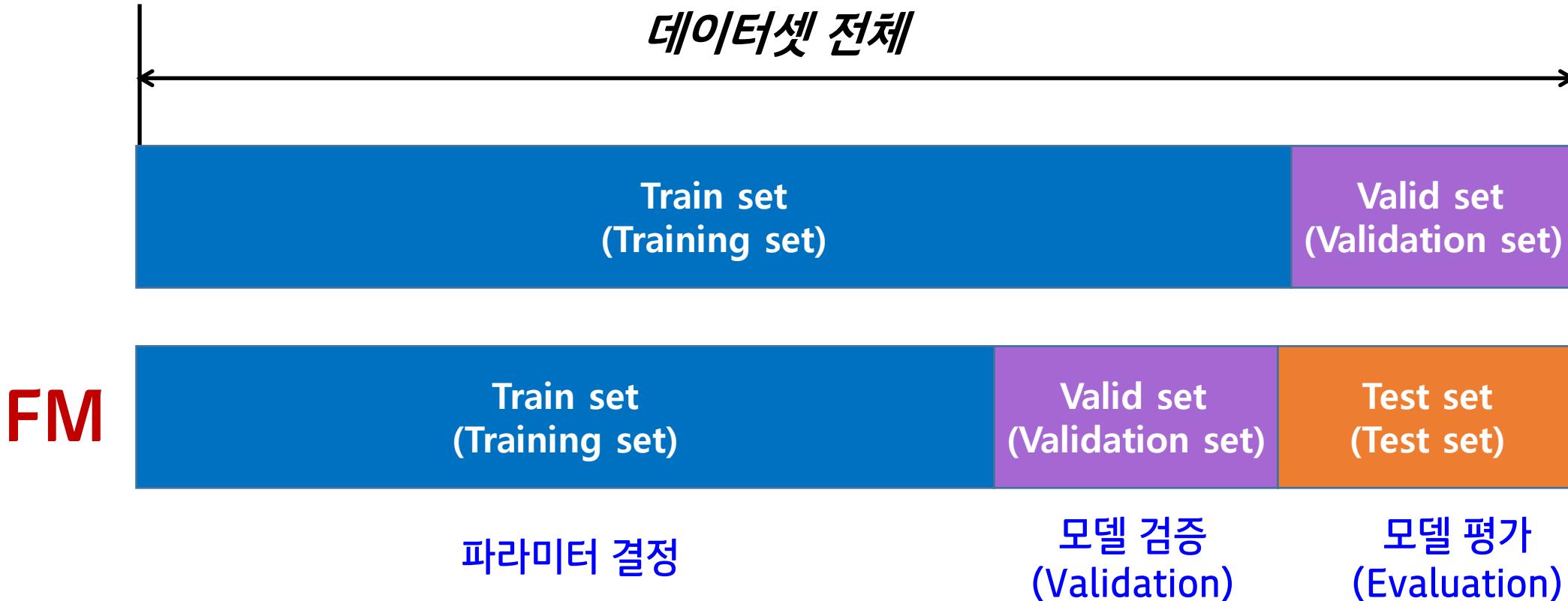
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

각 행(row)을  
예제(Example)  
데이터라고  
합니다

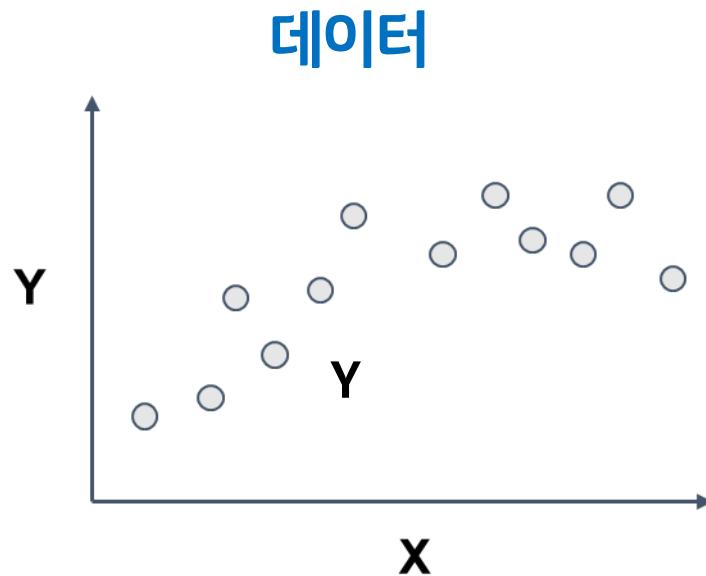
예측 모델(회귀 모델)  
팁의 크기를 예측

분류 모델  
손님의 성별을 예측

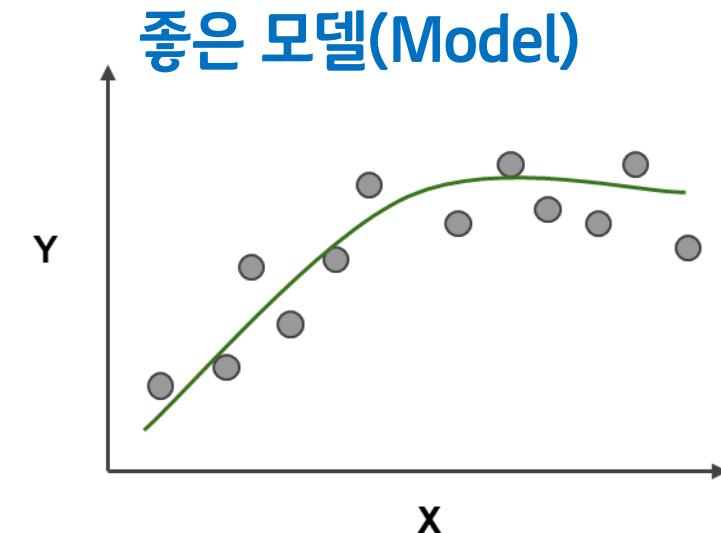
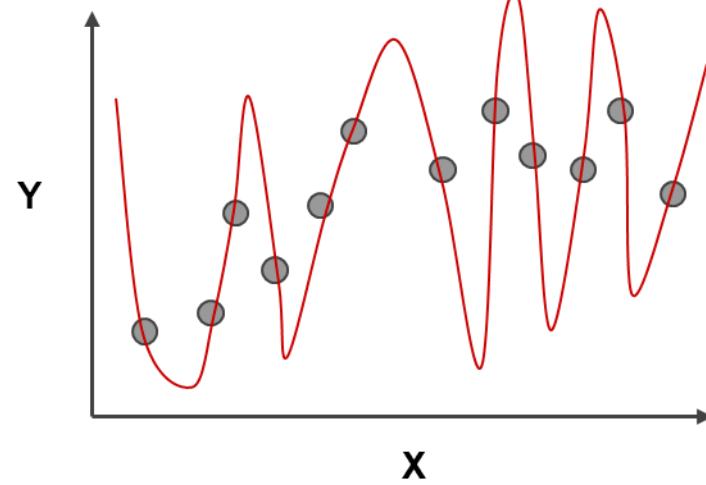
# 데이터셋 분리



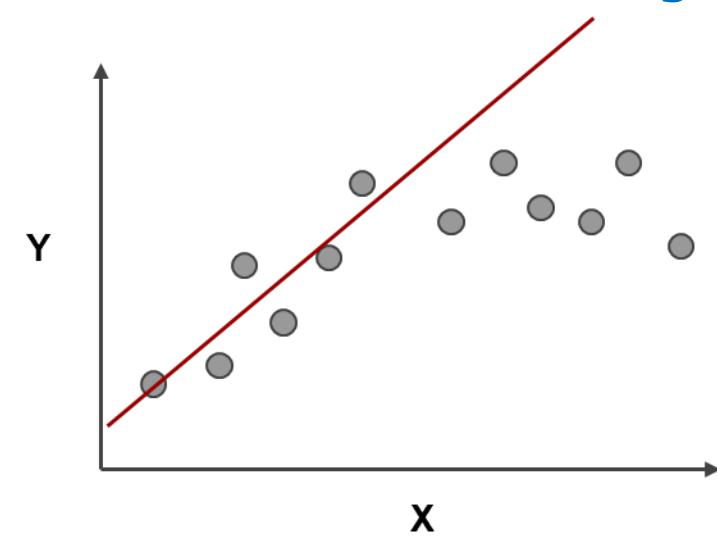
# 모델 선택



과적합(Overfitting)

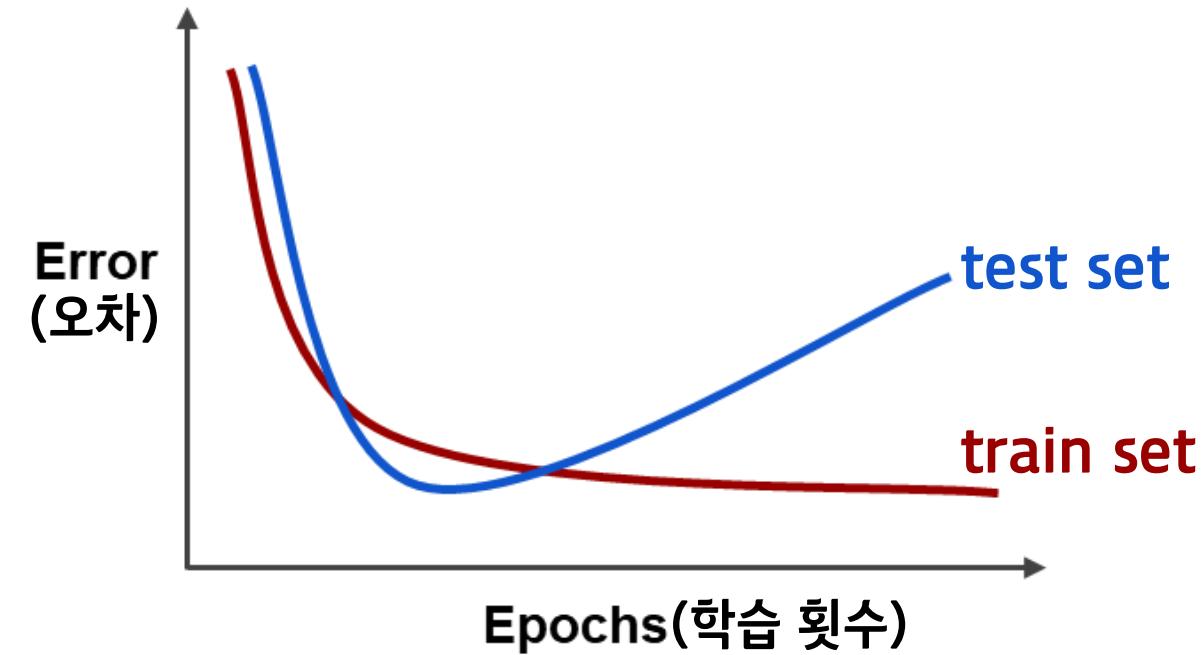
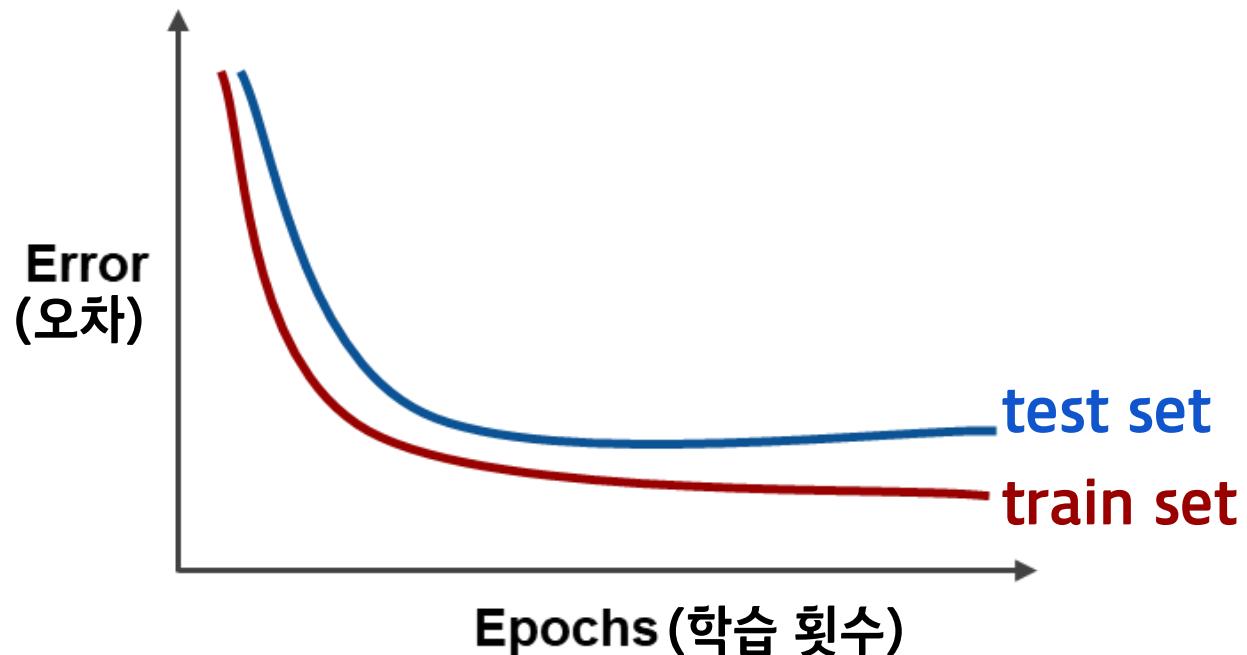


과소적합(Underfitting)



# 모델 성능 측정

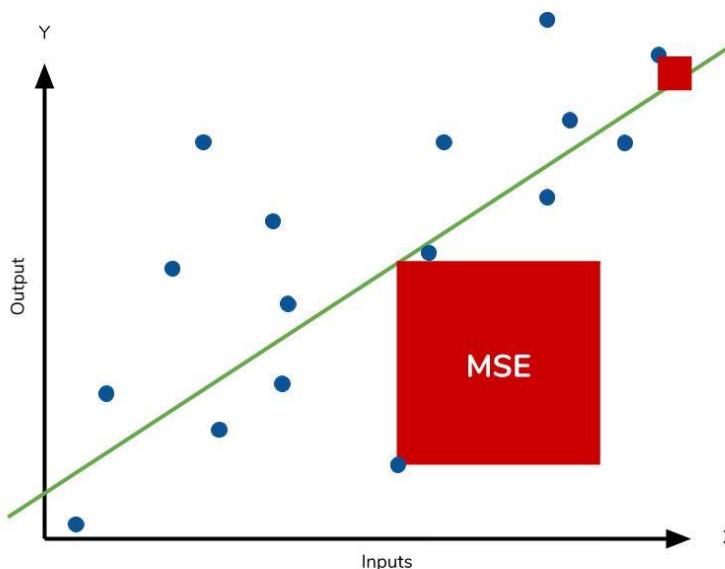
모델학습을 진행하면서 손실(Loss, Error, 오차)을 지속적으로 측정합니다.



# 회귀모델 손실함수(Loss Function)

회귀모델(Regression)에서는 주로 평균제곱오차(MSE)를 손실함수로 사용합니다.

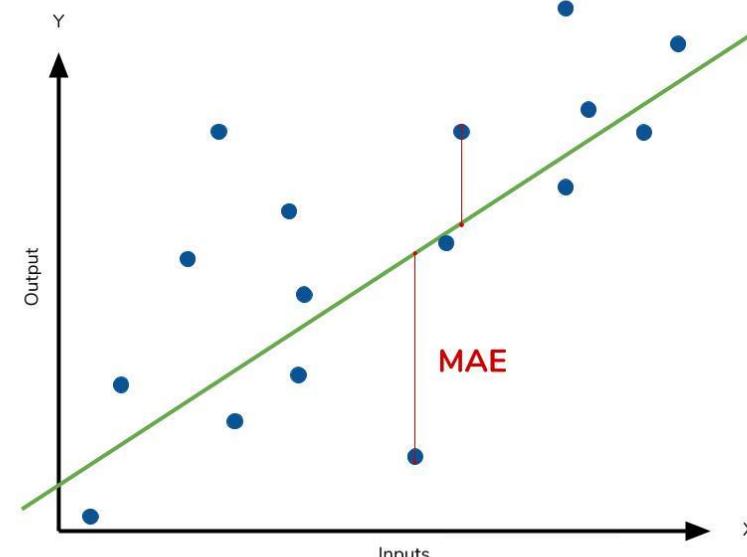
## ■ MSE(Mean Squared Error)



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

레이블 값  
(실제값)  
 $y_{\text{hat}}$   
(모델이 예측한 값)

## ■ MAE(Mean Absolute Error)



$$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

# 분류모델 손실함수(Loss Function)

이진분류는 Binary Cross Entropy 를 다중분류는 Categorical Cross Entropy 손실함수를 사용합니다.

## ■ 이진분류(Binary Classification)

0



1



## ■ 다중분류(Multi-Class Classification)

$C = 3$

Samples



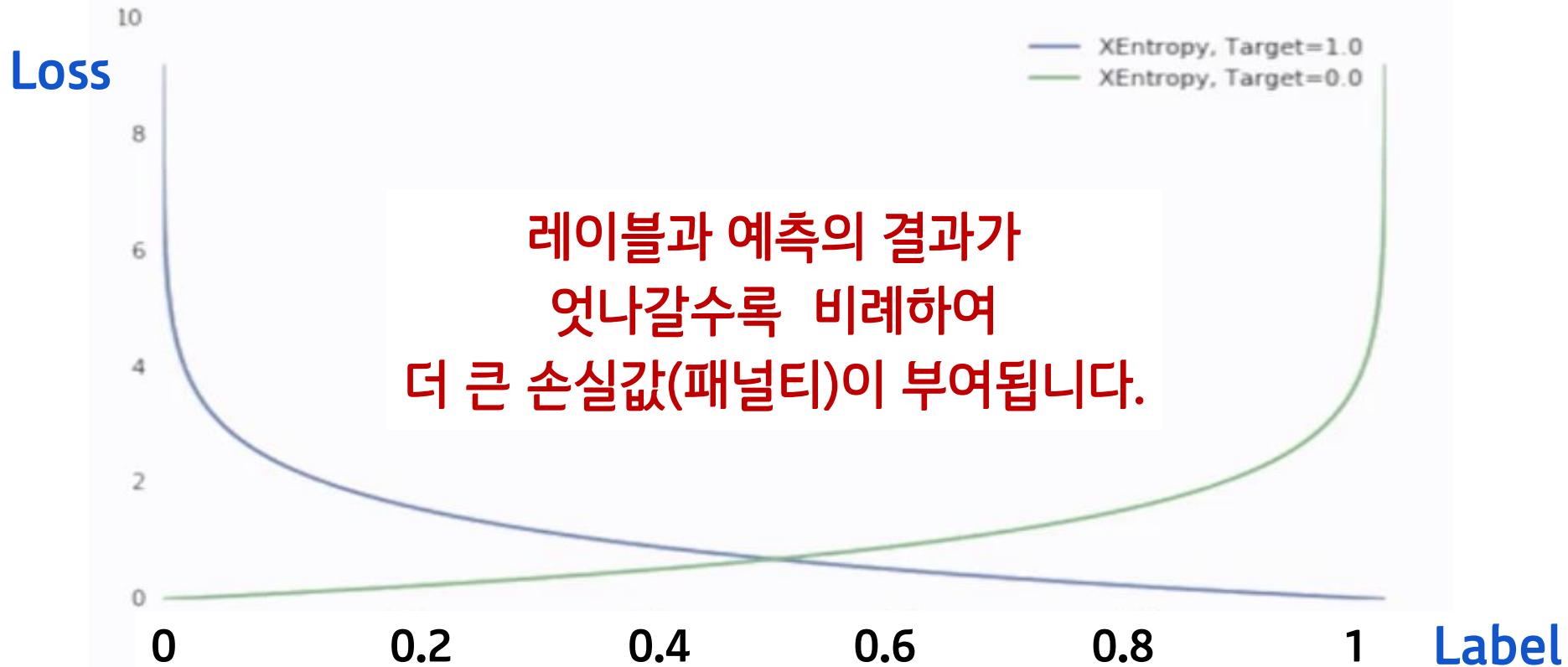
Labels ( $t$ )

[0 0 1]

[1 0 0]

[0 1 0]

# 분류모델 손실함수(Loss Function)



$$\frac{-1}{N} \times \sum_{i=1}^N y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i)$$

참고자료 : <https://youtu.be/Jt5BS71uVfl>

<http://kocw-n.xcache.kincdn.com/data/document/2017/kumoh/kojaepil0302/8.pdf>

# 오차 행렬(Confusion Matrix)

분류모델 성능평가에 사용하며 대략적인 성능확인과 모델의 성능을 오차행렬을 기반으로 수치로 표현할 수 있습니다.

		실제값	
		Positive	Negative
예측값	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Positive = True = 1

Negative = False = 0

TP(True Positive) : Positive(값 1)로 예측했는데, 실제값 역시 Positive(값 1)

TN(True Negative) : Negative(값 0)으로 예측는데, 실제값 역시 Negative(값 0)

FP(False Positive) : Positive(값 1)로 예측 했는데, 실제값은 Negative(값 0)

FN(False Negative) : Negative(값 0)으로 예측했는데, 실제 값은 Positive(값 1)

# 모델성능 평가지표

## ■ 정밀도(Precision)

모델이 True(Positive) 라고 분류한 것 중에서 실제 True(Positive) 인 것의 비율  
날씨 예측 모델이 맑다로 예측했는데, 실제 날씨가 맑았는지는 나타낸 지표입니다.

$$(Precision) = \frac{TP}{TP + FP}$$

## ■ 재현율/회수율(Recall)

실제 True인 것 중에서 모델이 True라고 예측한 것의 비율  
실제 날씨가 맑은 날 중에서 모델이 맑다고 예측한 비율을 나타낸 지표입니다.

$$(Recall) = \frac{TP}{TP + FN}$$

## ■ 정확도(Accuracy)

가장 직관적으로 모델의 성능을 나타낼 수 있는 평가 지표  
혼동행렬상에서는 대각선(TP)을 전체 셀로 나눈 값에 해당합니다.  
한달 동안에 맑은 날이 28일이고 비가 오는 날이 이틀인 경우, 비가 오는 것을  
예측하는 성능은 매우 낮을 수 밖에 없으므로 이를 보완할 지표가 필요합니다.

$$(Accuracy) = \frac{TP + TN}{TP + FN + FP + TN}$$

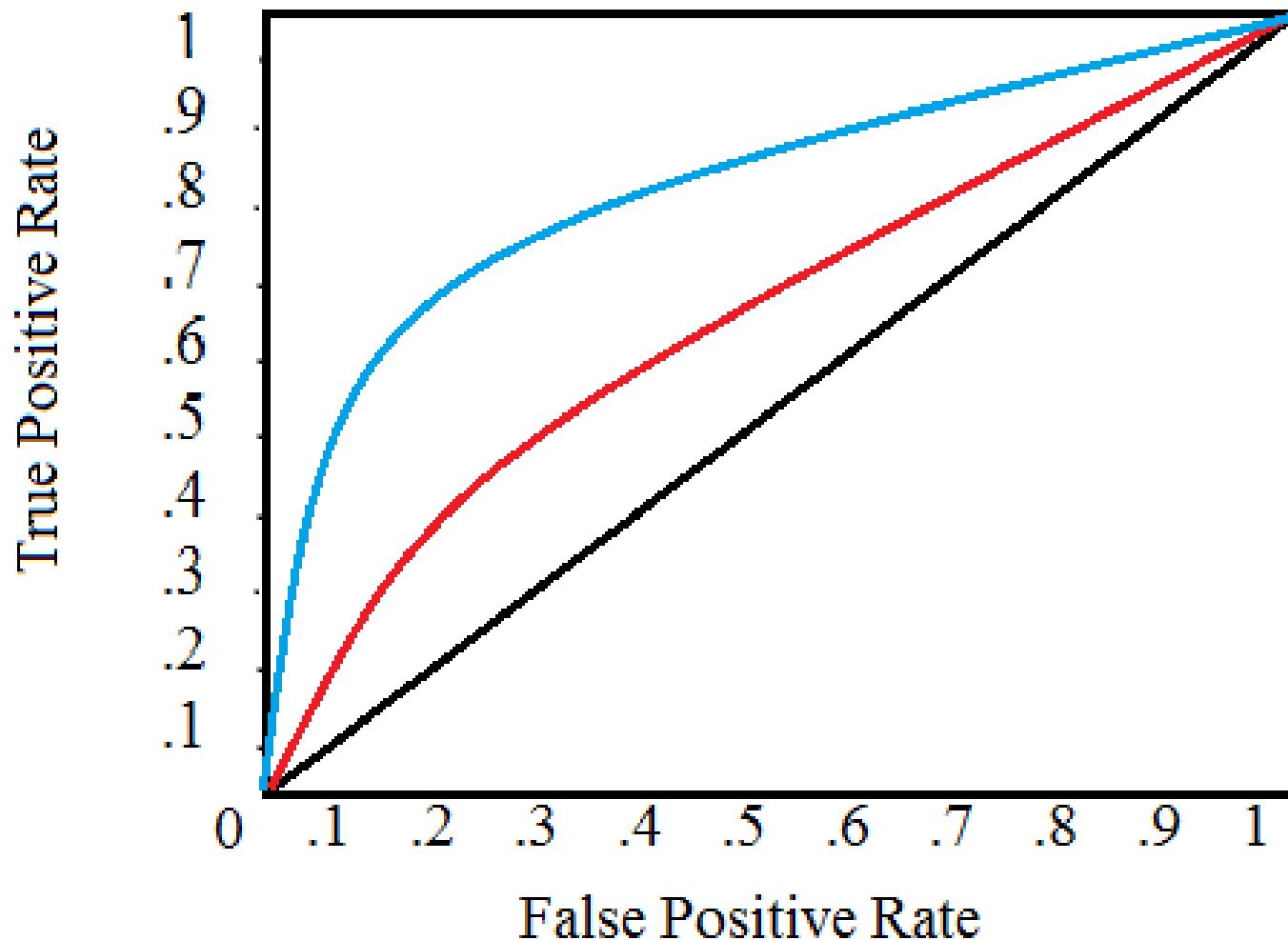
## ■ F1 점수(F1 score)

정밀도와 재현율의 조화평균입니다.

$$(F1-score) = 2 \times \frac{\frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

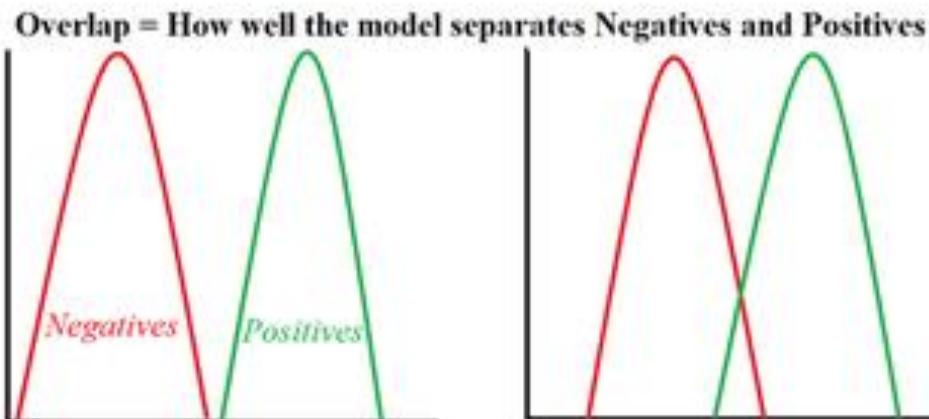
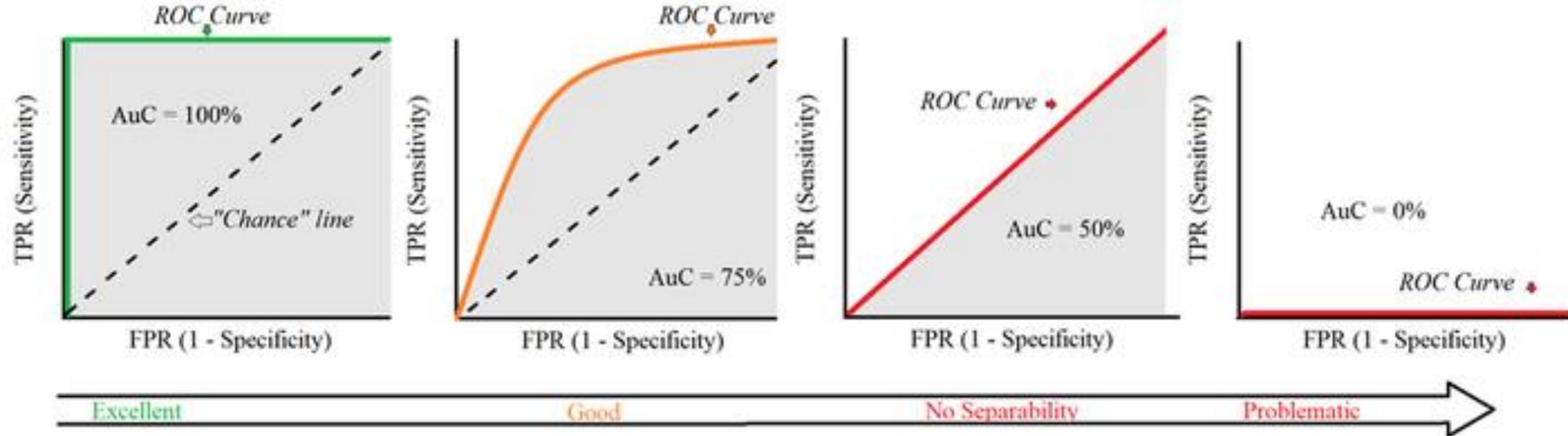
# ROC 곡선(Receiver Operating Characteristic)

거짓 양성 비율(False Positive Rate)에 대한 진짜 양성 비율(True Positive Rate, 재현율)의 곡선입니다.  
ROC 곡선을 통해 모델의 성능을 평가하거나 최적의 분류기준(threshold)을 찾을 수 있습니다.

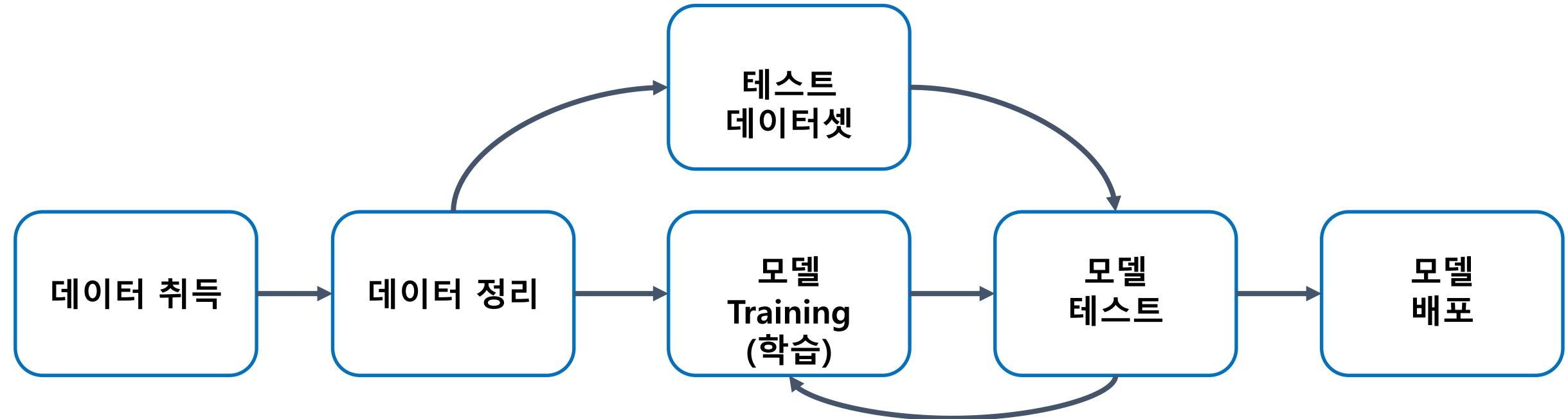


# AUC(Area under the Curve)

ROC 곡선 아래의 면적(AUC)을 측정하면 모델의 성능을 평가하거나 최적의 분류기준(threshold)을 찾을 수 있습니다.



# 머신러닝 프로세스



# 사이킷런(Scikit-learn)

가장 인기있는 머신러닝 패키지이며, 많은 머신러닝 알고리즘이 내장되어 있습니다.

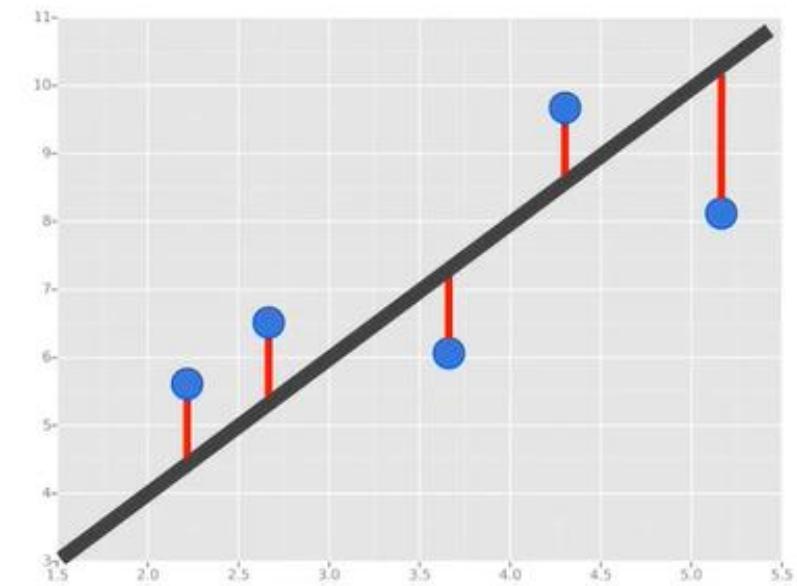
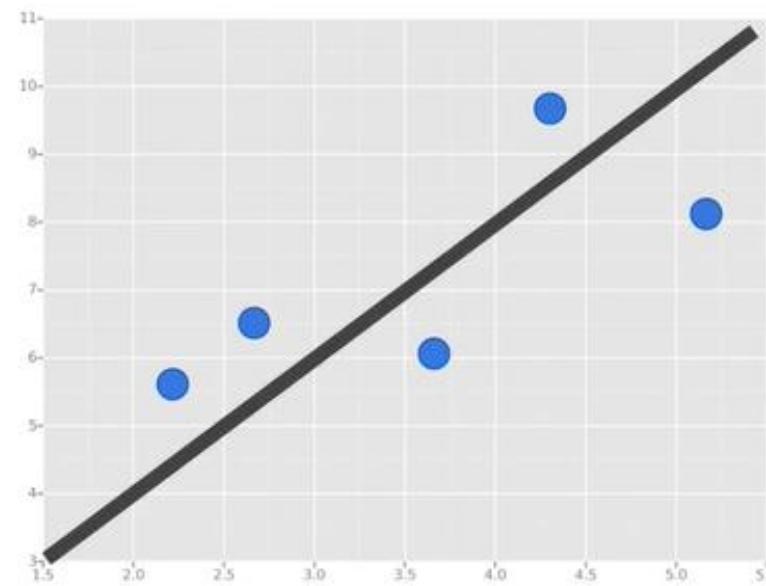
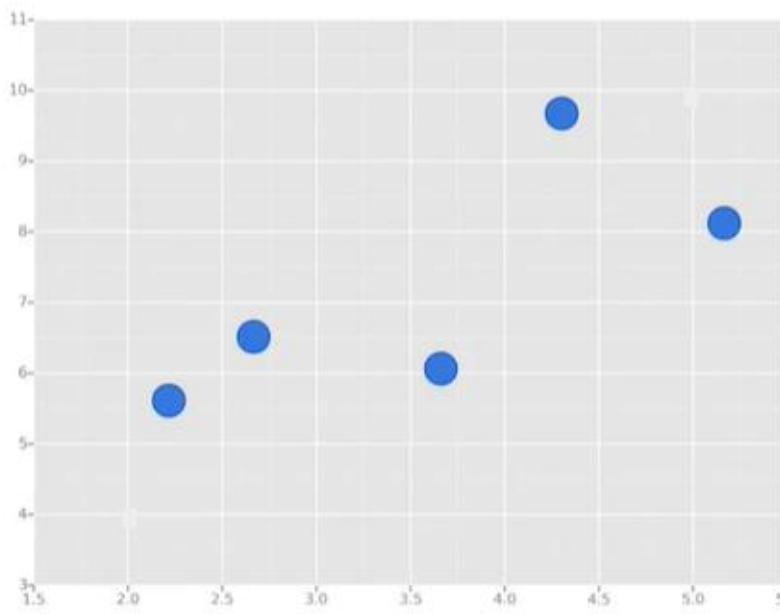
The screenshot shows the official website for scikit-learn (<https://scikit-learn.org>). The top navigation bar includes links for 'Install', 'User Guide', and 'API'. Below the header, there's a main title 'scikit-learn' and a subtitle 'Machine Learning in Python'. A blue banner highlights the following features:

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

The page is divided into three main sections: 'Classification', 'Regression', and 'Clustering'. Each section contains a brief description, applications, algorithms, and a corresponding visualization.

- Classification:** Identifying which category an object belongs to. Applications include spam detection and image recognition. Algorithms include SVM, nearest neighbors, random forest, and more. It features a grid of 9x6 plots comparing various classification models like K-Nearest Neighbors, Logistic Regression, and Decision Trees against a baseline.
- Regression:** Predicting a continuous-valued attribute associated with an object. Applications include drug response and stock prices. Algorithms include SVR, nearest neighbors, random forest, and more. It shows a plot of 'Boosted Decision Tree Regression' with multiple fitted curves for different numbers of estimators.
- Clustering:** Automatic grouping of similar objects into sets. Applications include customer segmentation and grouping experiment outcomes. Algorithms include k-Means, spectral clustering, mean-shift, and more. It displays a scatter plot of digits data points grouped into four clusters with their respective centroids marked by white crosses.

# 선형 회귀(Linear Regression)



# 선형 회귀(Linear Regression)



06\_01\_LinearRegression.ipynb

```
from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression  
  
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]).reshape(-1,1)  
y = np.array([13, 25, 34, 47, 59, 62, 79, 88, 90, 100])  
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                    test_size=0.3, random_state=42)  
  
model = LinearRegression()  
  
model.fit(X_train, y_train)  
  
predictions = model.predict(X_test)
```

# 분류(Classification)



08\_Classification.ipynb

■ setosa



■ versicolor



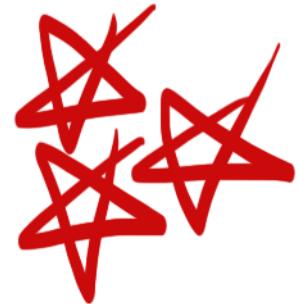
■ virginica



```
from sklearn import datasets  
from sklearn.model_selection import train_test_split  
iris = datasets.load_iris()
```

# Train Test 데이터셋 분할

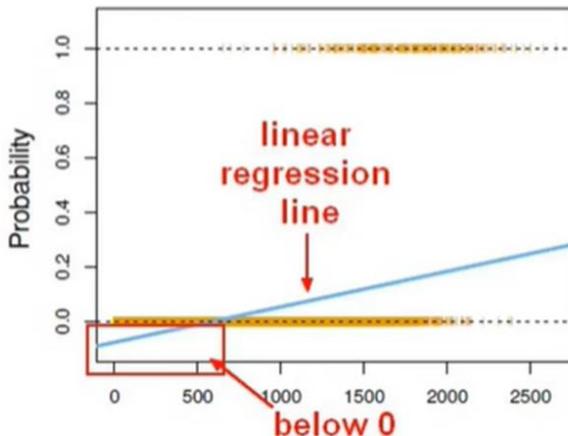
```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(  
    iris['data'],  
    iris['target'],  
    test_size=0.3,  
    shuffle=True,  
    stratify=iris.target,  
    random_state=42)
```



# 로지스틱 회귀(Logistic Regression)

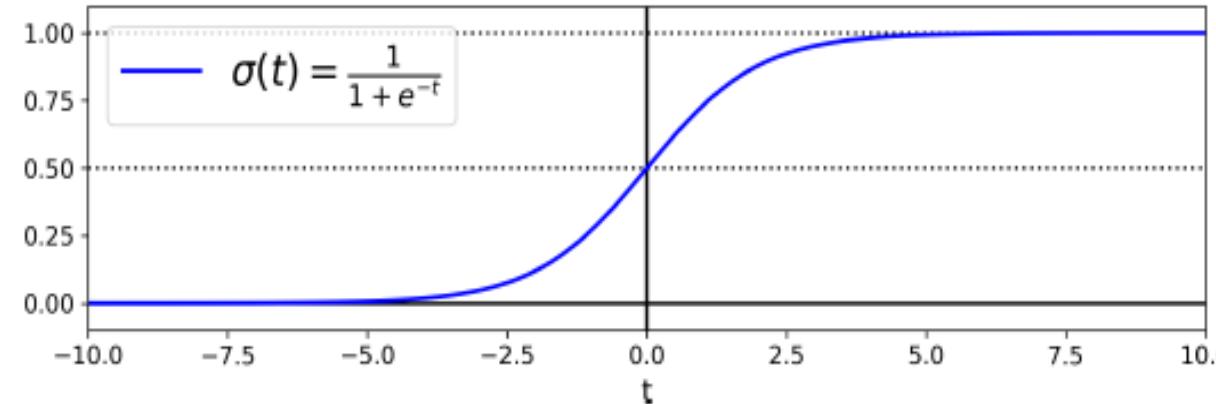
- 이진 분류 규칙은 0과 1의 두 클래스를 갖는 것으로, 일반 선형 회귀 모델을 이진분류에 사용할 수 없습니다.
- 대신 선형 회귀를 로지스틱 회귀 곡선으로 변환 할 수 있으며, 로지스틱 회귀 곡선은 0과 1 사이에서만 이동할 수 있으므로 분류에 사용할 수 있습니다.
- 로지스틱 회귀는 선형 회귀처럼 바로 결과를 출력하지 않고 로지스틱(logistic)을 출력합니다.
- 로지스틱 회귀는 샘플이 특정 데이터에 속할 확률을 추정(이진분류)하는 데 사용됩니다.
- 추정 확률이 50%가 넘으면 모델은 그 샘플이 해당 클래스에 속한다고 예측합니다.

일반 선형 모델



로지스틱 함수

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



로지스틱 확률모델

$$\hat{p} = h_{\theta}(x) = \sigma(x^T \theta)$$

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

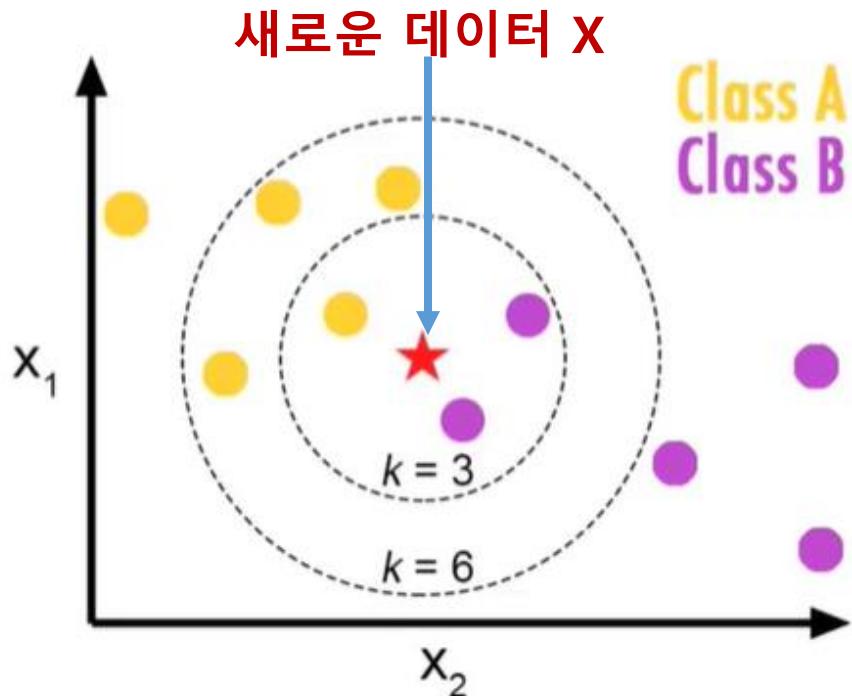
# 로지스틱 회귀(Logistic Regression)

```
from sklearn.linear_model import LogisticRegression
# 모델 학습
lr = LogisticRegression()
lr.fit(X_train, y_train)
# 예측
pred = lr.predict(X_test)
print ('예측값: ', pred[:10])

# 모델 성능 평가
accuracy = accuracy_score(y_test, pred)
print(f'Mean accuracy score: {accuracy:.4}')
# 확률값
prob = lr.predict_proba(X_test)
print("Probability: ", prob[0])
```

# KNN(K-Nearest Neighbor)

- KNN은 새로운 데이터가 주어졌을 때 기존 데이터 가운데 가장 가까운 k개 이웃의 정보로 새로운 데이터를 예측하는 방법론입니다. 아래 그림처럼 검은색 점의 범주 정보는 주변 이웃들을 가지고 추론해낼 수 있습니다.
- 만약 k값이 3이면 Class B, k가 6이면 Class A로 분류(classification)하는 것입니다.
- 만약, 회귀(regression) 문제라면 이웃들 종속변수(y)의 평균이 예측값이 됩니다.
- 알고리즘이 간단하며 큰 데이터셋과 고차원 데이터에 적합하지 않은 단점이 있습니다.



K	이웃(Neighbor)	예측값
3		Class B
7	 	Class A

# KNN(K-Nearest Neighbor)

```
from sklearn.linear_model import KNeighborsClassifier
```

# 모델 학습

```
knn = KNeighborsClassifier(n_neighbors=7)  
knn.fit(X_train, y_train)
```

# 예측

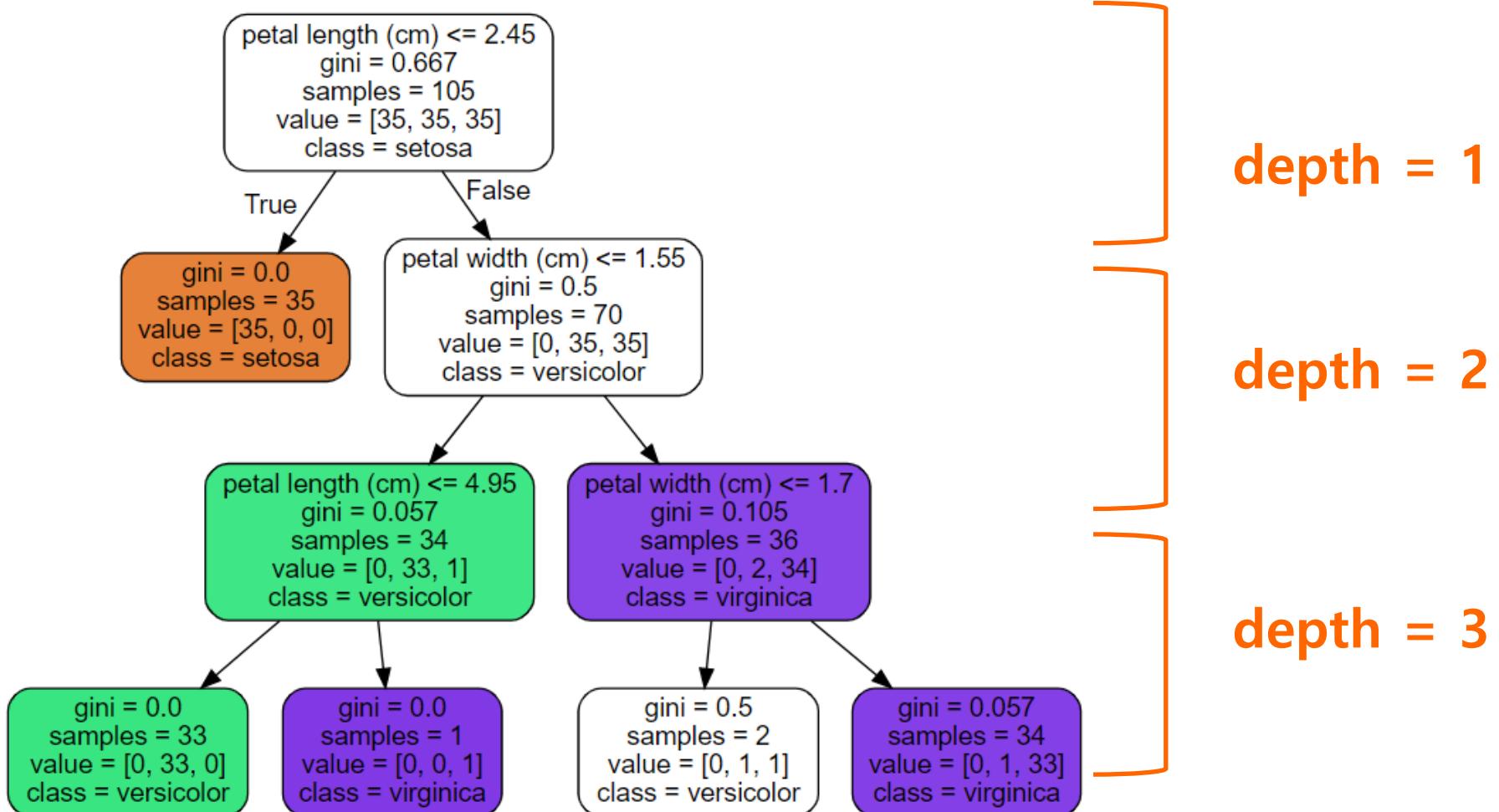
```
pred = knn.predict(X_test)  
print ('예측값: ', pred[:10])
```

# 모델 성능 평가

```
accuracy = accuracy_score(y_test, pred)  
print(f'Mean accuracy score: {accuracy:.4}')
```

# 의사결정트리(Decision Tree)

의사결정트리 모델은 트리(Tree) 알고리즘을 사용합니다. 트리의 각 분기점(node)에 데이터셋의 Feature를 하나씩 위치시키고, 각 분기점(node)에서 임의의 조건식으로 가지를 나무면서 데이터를 구분합니다.



# 의사결정트리(Decision Tree)

```
from sklearn.linear_model import DecisionTreeClassifier

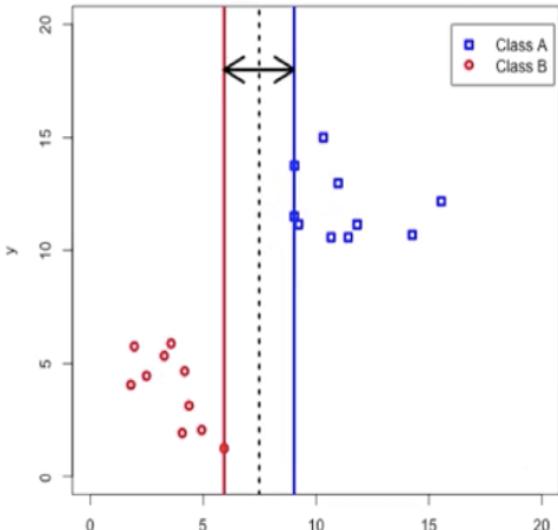
# 모델 학습
dtc = DecisionTreeClassifier(max_depth=3, random_state=42)
dtc.fit(X_train, y_train)
# 예측
pred = dtc.predict(X_test)
print('예측값: ', pred[:10])
# 모델 성능 평가
accuracy = accuracy_score(y_test, pred)
print(f'Mean accuracy score: {accuracy:.4f}')
# 확률값
prob = dtc.predict_proba(X_test)
print("Probability: ", prob[0])
```

# 서포트 벡터 머신(SVM)

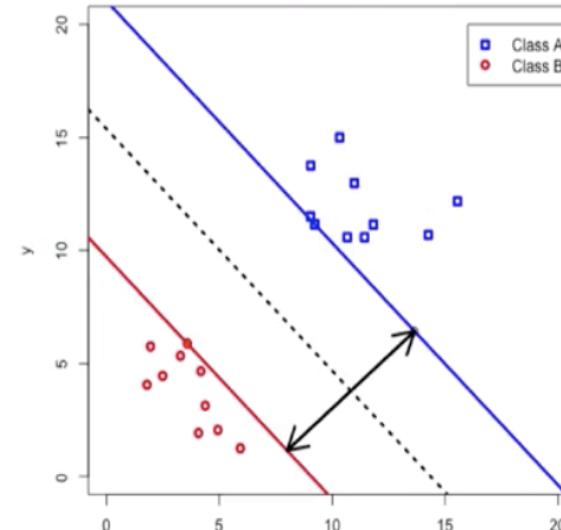
- 서포트 벡터 머신은 선형/비선형 분류, 회귀, 이상치 탐색에도 사용할 수 있는 다목적 머신러닝 모델입니다.
- SVM은 복잡한 분류 모델에 잘 들어 맞으며 작거나 중간 크기의 데이터셋에 적합합니다.

SVMs maximize the margin between two classes

Small margin



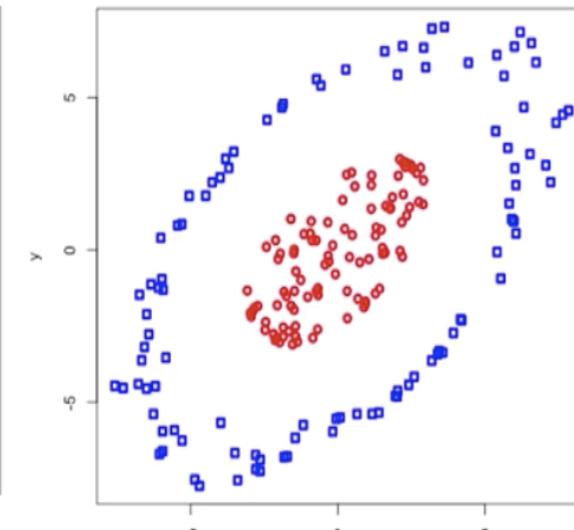
Large margin



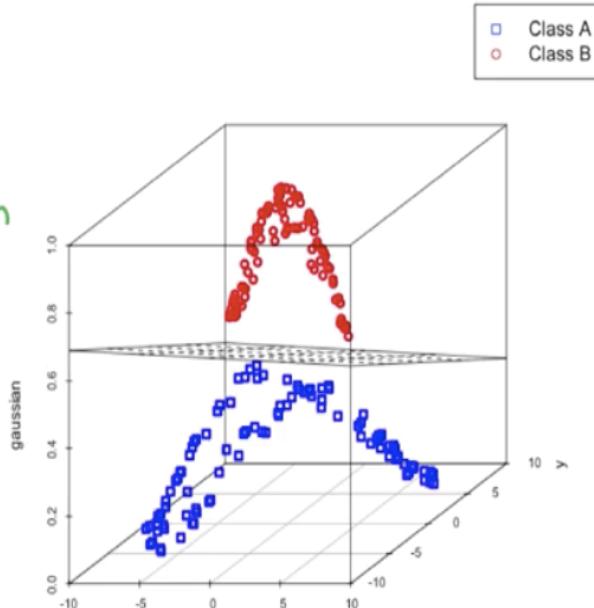
Kernels transform the input space into a more usable feature space

Class A  
Class B

Class A  
Class B



Gaussian  
RBF  
Kernel



# ■ 서포트 벡터 머신(SVM)

```
from sklearn.svm import SVC
```

```
# 모델 학습
```

```
svc = SVC(kernel='rbf')  
svc.fit(X_train, y_train)
```

```
# 예측
```

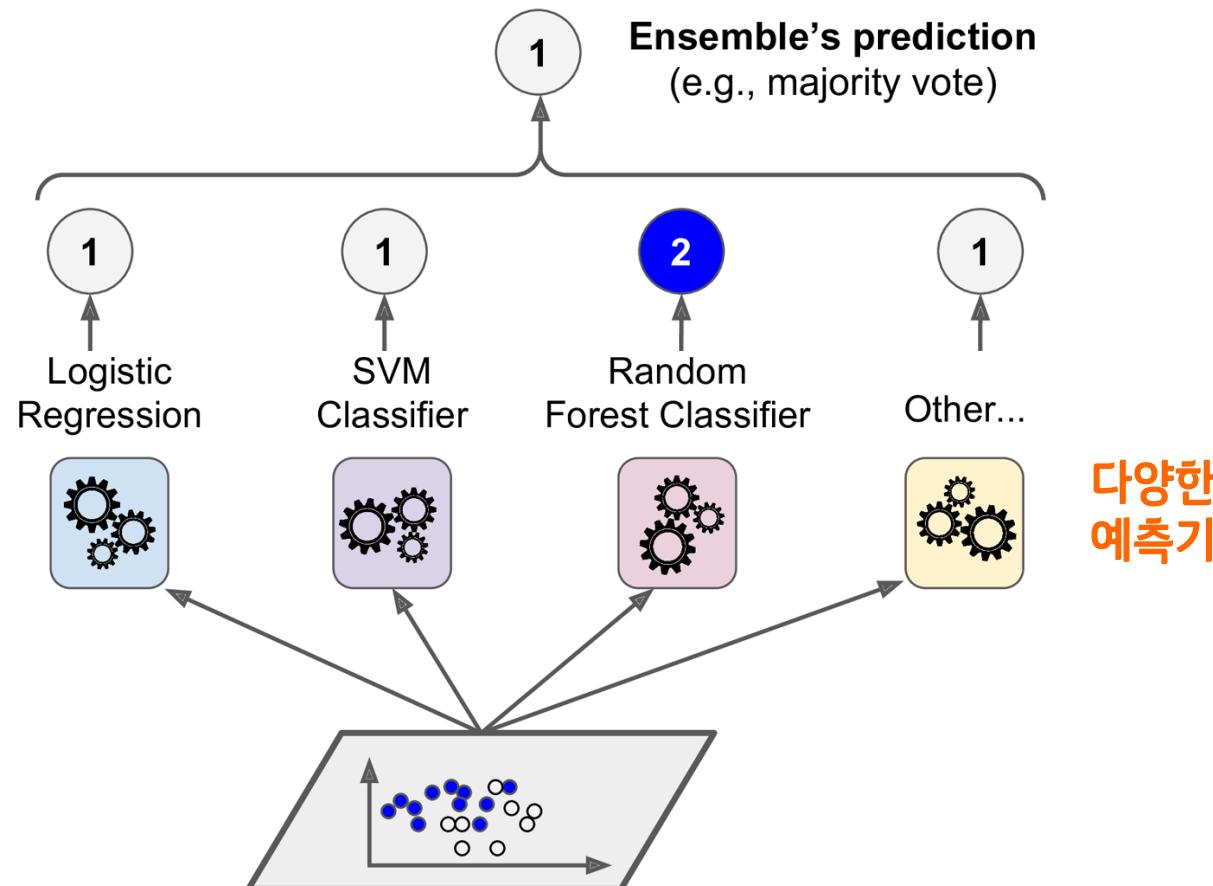
```
pred = svc)  
print ('예측값: ', pred[:10])
```

```
# 모델 성능
```

```
acc = accuracy_score(y_test, pred)  
print("Accuracy: {:.4f}".format(acc))
```

# 앙상블 학습(Ensemble Learning)

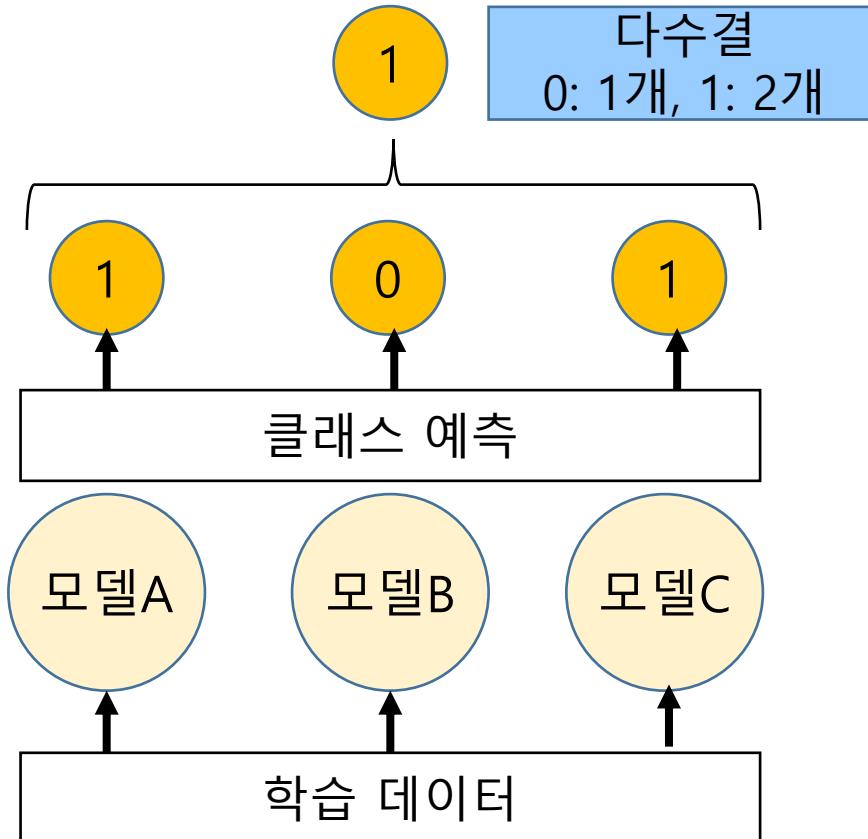
- 일련의 예측기(분류, 회귀)로부터 예측을 수집하면, 가장 좋은 모델 1개보다 더 좋은 예측을 얻을 수 있을 것입니다.
- 일련의 예측기를 앙상블이라 부르고 이를 앙상블 학습(Ensemble Learning)이라고 합니다.
- 가장 인기 있는 앙상블 방법에는 배깅, 부스팅이 있습니다.



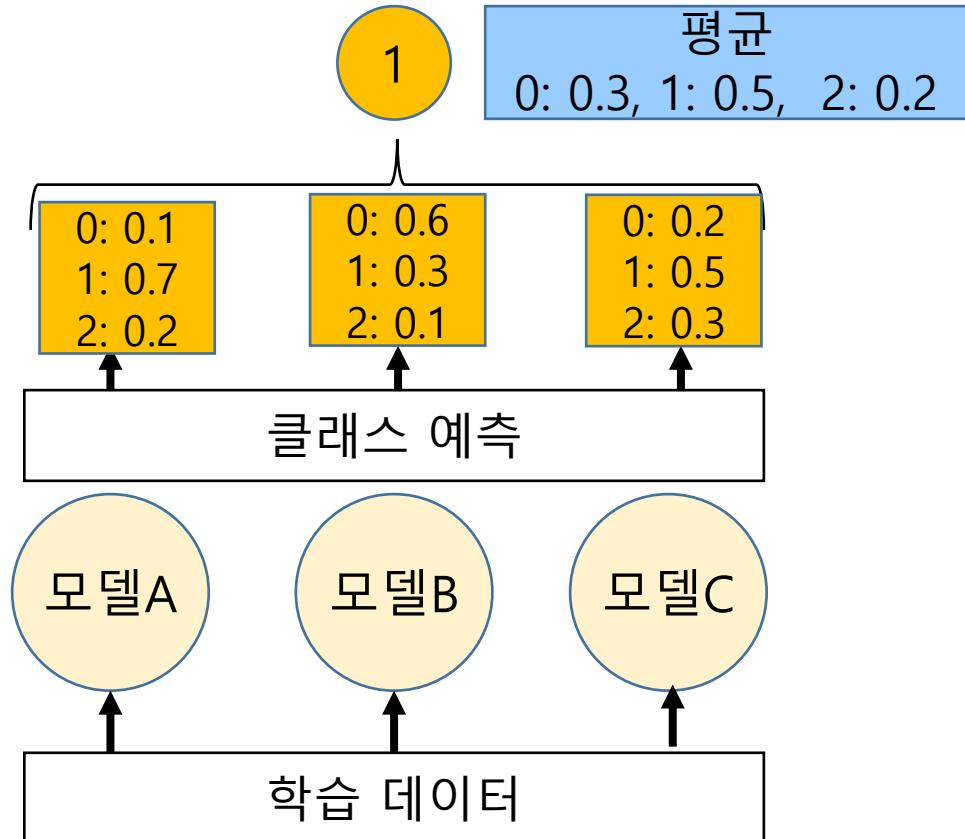
# 앙상블 모델 - 보팅(Voting)

- 보팅(Voting)은 여러 개의 모델이 예측한 값을 결합하여 최종 예측값을 결정하는 앙상블 방법입니다.
- 하드 보팅(hard voting)은 모델이 예측한 값 중에서 다수결로 최종 분류 클래스를 정합니다.
- 소프트 보팅(soft voting)은 각 분류 클래스별 예측 확률을 평균하여 최종 분류 클래스를 정합니다.

## ■ 하드 보팅(hard voting)



## ■ 소프트 보팅(soft voting)



# 앙상블 모델 - 보팅(Voting)

```
from sklearn.ensemble import VotingClassifier
```

```
# 모델 학습
```

```
hvc = VotingClassifier(estimators=[('KNN', knn), ('DT', dtc),
('SVM', svc)], voting='hard')
```

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

```
# 예측
```

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

```
print('예측값:', pred[:10])
```

```
# 모델 성능 평가
```

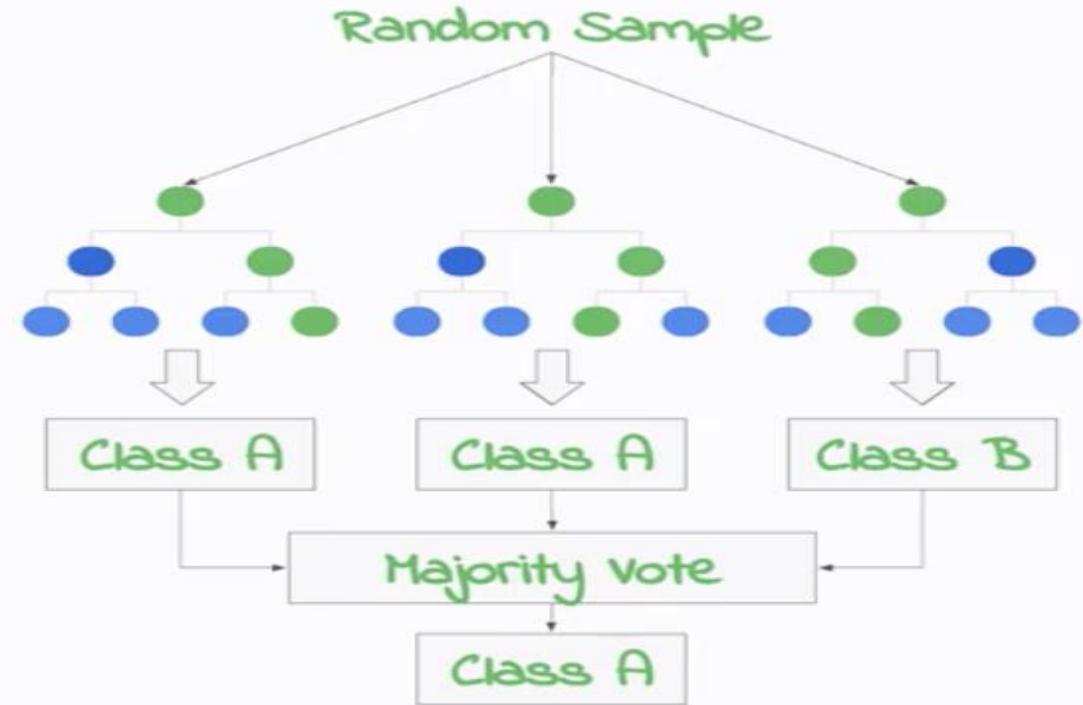
```
acc = accuracy_score(y_test, pred)
```

```
print("Accuracy: {:.4f}".format(acc))
```

# 앙상블 모델 - 배깅(Bagging)

- 다양한 분류기를 만드는 각기 다른 훈련 알고리즘을 사용하는 것과, 같은 알고리즘을 사용하고, 훈련 세트의 서브셋을 무작위로 구성하여 각기 다르게 학습시키는 방법이 있습니다.
- 훈련세트에서 중복을 허용하여 샘플링 하는 방식을 **bootstrap aggregating**, 배깅(**bagging**)라고 합니다.
- 통계학에서 중복을 허용한 리샘플링을 부트스트래핑(bootstrapping)이라고 합니다.
- 중복을 허용하지 않고 샘플링 하는 방식은 페이스팅(pasting)이라고 합니다.
- 랜덤 포레스트(Random Forest)는 일반적으로 배깅(또는 페이스팅)을 적용한 의사결정트리의 앙상블입니다.

Random forest: Strong learner from many weak learners



# 앙상블 모델 - 랜덤 포레스트(Random Forest)



```
from sklearn.ensemble import RandomForestClassifier
```

```
# 모델 학습
```

```
rfc = RandomForestClassifier(n_estimators=50, max_depth=3,  
                            random_state=20)
```

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

```
# 예측
```

```
pred = rfc.predict(X_test)  
print('예측값:', pred[:10])
```

```
# 모델 성능 평가
```

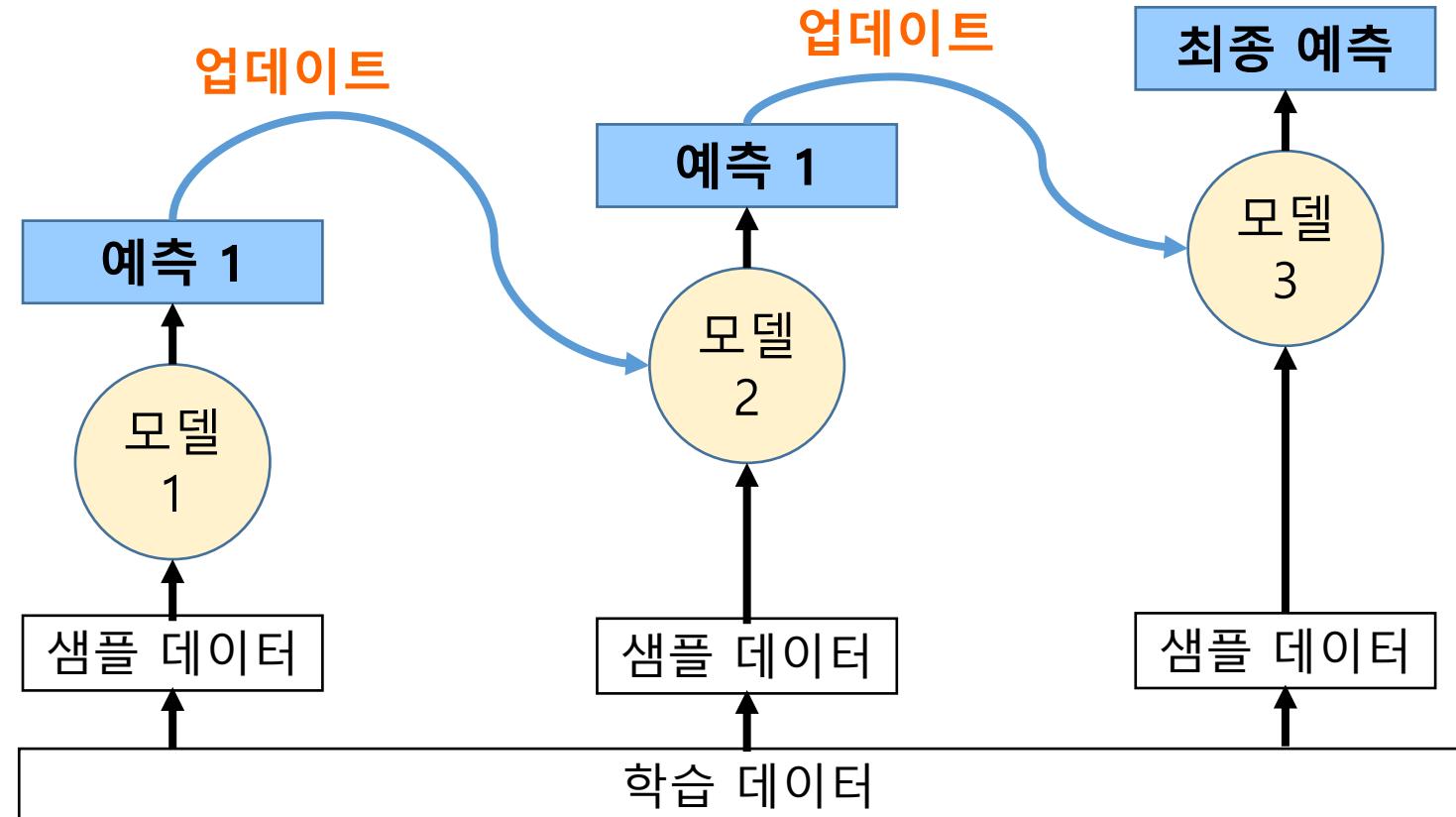
```
accuracy = accuracy_score(y_test, pred)  
print(f'Mean accuracy score: {accuracy:.4f}')
```

shift+tab키 : 함수 설명 보기

```
Init signature:  
RandomForestClassifier(  
    n_estimators=100,  
    *,  
    criterion='gini',  
    max_depth=None,  
    min_samples_split=2,  
    min_samples_leaf=1,  
    min_weight_fraction_leaf=0.0,  
    max_features='auto',
```

# 앙상블 모델 - 부스팅(Boosting)

- 부스팅(Boosting)은 여러 개의 모델을 순차적으로 학습합니다.
- 잘못 예측한 데이터에 대한 예측 오차를 줄일 수 있는 방향으로 모델을 계속 업데이트 합니다.
- XGBoost 모델은 Kaggle(<https://www.kaggle.com/>) 경진대회에서 많이 사용되고 있는 알고리즘의 하나입니다.



# 앙상블 모델 - 부스팅(XGBoost)



```
!pip install xgboost
```

```
from xgboost import XGBClassifier
```

```
# 모델 학습
```

```
xgbc = XGBClassifier(n_estimators=50, max_depth=3,  
                      random_state=42)
```

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

```
# 예측
```

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

```
print('예측값: ', pred[:10])
```

```
# 모델 성능 평가
```

```
acc = accuracy_score(y_test, pred)
```

```
print(f'Mean accuracy score: {accuracy:.4}')
```

# 머신러닝 모델구현 실습



데이터파일 : churn\_data.csv

customerID	gender	SeniorCitizen	TotalCharges	Churn
7590-VHVEG	Female	0	29.85	No
5575-GNVDE	Male	0	1889.5	No
3668-QPYBK	Male	0	108.15	Yes
7795-CFOCW	Male	0	1840.75	No
9237-HQITU	Female	0	151.65	Yes
9305-CDSKC	Female	0	820.5	Yes
1452-KIOVK	Male	0	1949.4	No
6713-OKOMC	Female	0	301.9	No

- customerID: 고객ID
- gender: 고객 성별
- SeniorCitizen: 고객이 노약자인가 아닌가
- Partner: 고객에게 파트너가 있는지 여부(결혼 여부)
- Dependents: 고객의 부양 가족 여부
- tenure: 고객이 회사에 머물렀던 개월 수
- PhoneService: 고객에게 전화 서비스가 있는지 여부
- MultipleLines: 고객이 여러 회선을 사용하는지 여부
- InternetService: 고객의 인터넷 서비스 제공업체
- OnlineSecurity: 고객의 온라인 보안 여부
- OnlineBackup: 고객이 온라인 백업을 했는지 여부
- DeviceProtection: 고객에게 기기 보호 기능이 있는지 여부
- TechSupport: 고객이 기술 지원을 받았는지 여부
- StreamingTV: 고객이 스트리밍TV를 가지고 있는지 여부
- StreamingMovies: 고객이 영화를 스트리밍하는지 여부
- Contract: 고객의 계약기간
- PaperlessBilling: 고객의 종이 없는 청구서 수신 여부(모바일 청구서)
- PaymentMethod: 고객의 결제 수단
- MonthlyCharges: 매월 고객에게 청구되는 금액
- TotalCharges: 고객에게 청구된 총 금액
- Churn: 고객 이탈 여부 (label, y)

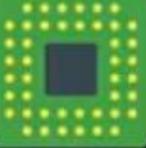
# 머신러닝 모델구현 실습



09-ML-Exercise.ipynb



1. TotalCharges 컬럼의 공백값을 문자 '0'으로 변경하고 수치형 데이터 타입으로 변환하세요.
2. 고객이탈여부 데이터를 변수 y에 할당하고 나머지 데이터를 변수 X에 할당하세요.
3. X, y 데이터셋을 70%:30% 비율로 훈련데이터셋과 검증데이터셋으로 분할하세요
4. 랜덤 포레스트 모델로 이탈고객 예측분류기를 만들고 모델성능을 출력하세요.
5. XGBoost 모델로 이탈고객 예측분류기를 모델성능을 측정하세요.



제프리 힌튼



훈련 데이터

출처 : <https://youtu.be/C2sqt9pG6K0>



# 5. 스타트 딥러닝

컴퓨터가 데이터를 이용해 마치 사람처럼 스스로 학습할 수 있도록

인공신경망을 기반으로 한 기계학습 기술이다.

딥러닝 기술을 적용하면 사람이 모든 판단 기준을 정해주지 않아도

컴퓨터가 스스로 인지·추론·판단할 수 있게 된다.

딥러닝은 학습을 위한 더 나은 표현 방법과 효율적인 모델 구축에 초점을 맞춰

진행이 되고 있으며 이러한 표현 방법들 중 다수는 신경과학에서 영감을 얻는 경우가 많다.

출처 : <https://bit.ly/39ImCQd>

인공신경망(Artificial Neural Network, ANN)

심층신경망(Deep Neural Network, DNN)

가중치(Weight)

활성화 함수(Activation Function)

손실함수(Loss Function)

딥러닝 학습방법(Deep Learning)

순전파(Forward Propagation)

오차역전파>Error Back Propagation)

경사하강법(Gradient Descent)

옵티마이저(Optimization Algorithm)

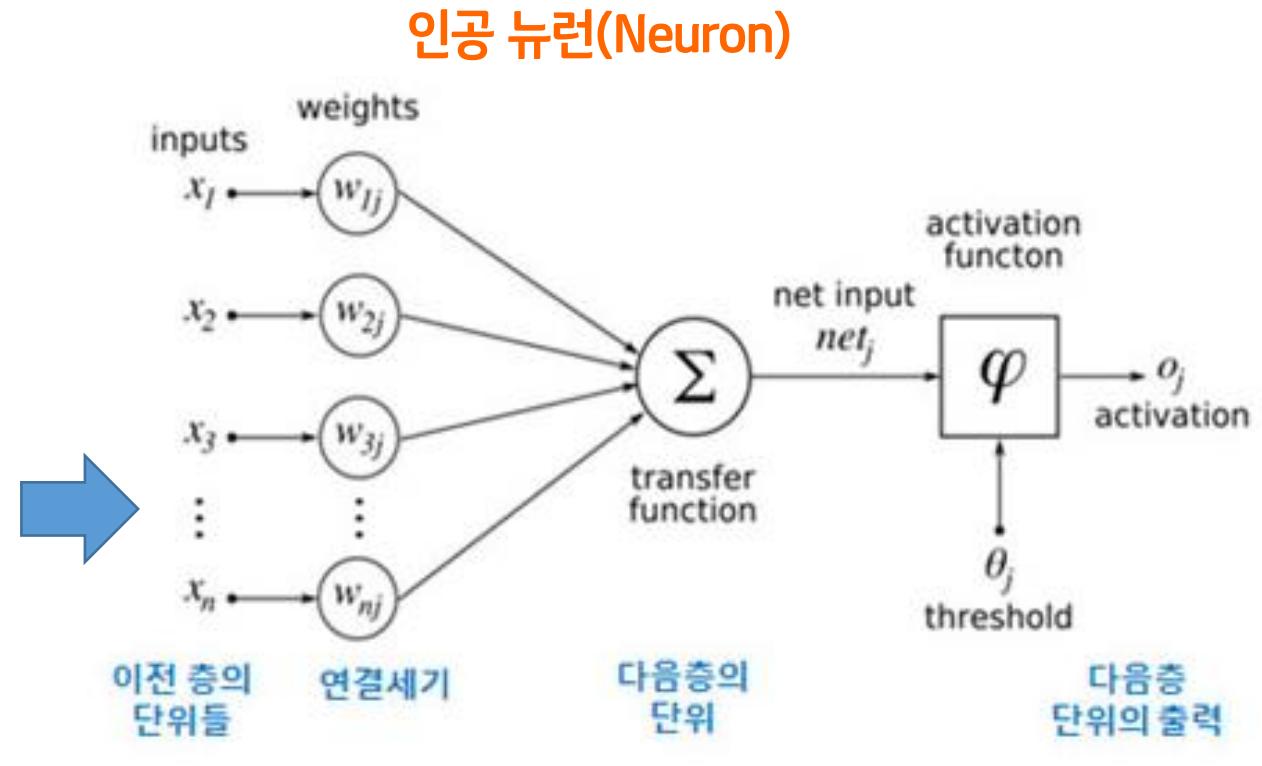
드롭아웃(Dropout)

미니배치(Mini Batch)

텐서(Tensor)

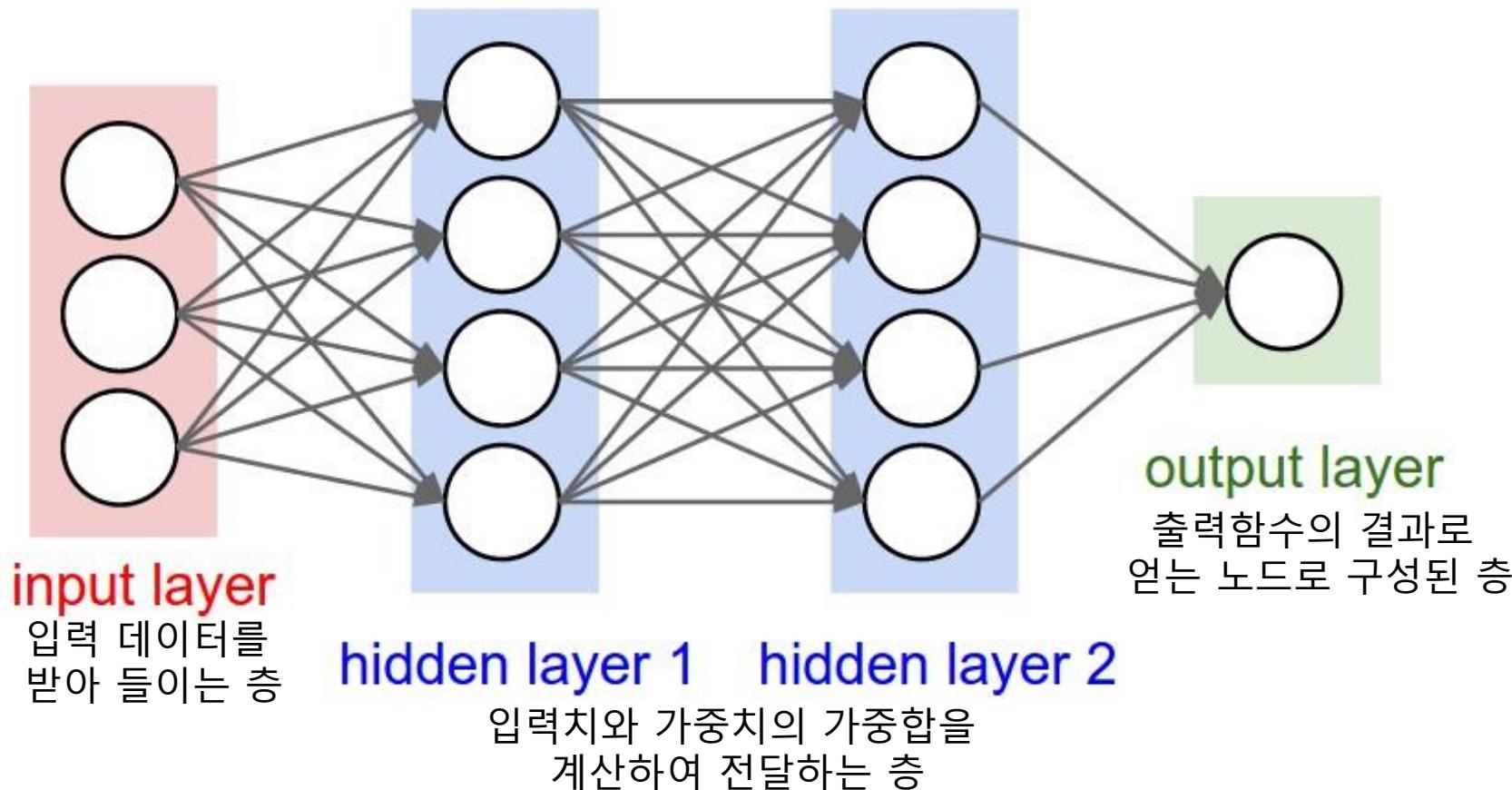
# 인공신경망(Artificial Neural Network, ANN)

- 뇌신경은 수많은 신경세포(뉴런, neuron)들이 연결되어 정보를 처리하고 전달합니다.
- 인공신경망은 뇌 신경계의 정보처리 구조를 모방하여 만든 계산 알고리즘입니다.
- 뇌 신경계와 같이 수많은 계산 함수를 연결하여 복잡한 정보를 처리하는 네트워크 구조입니다.



# 심층신경망(Deep Neural Network, DNN)

- 딥러닝은 여러 층(layer)을 가진 인공신경망(Artificial Neural Network, ANN)을 사용하여 학습을 수행하는 것입니다.
- 심층신경망은 입력층과 출력층사이에 다수의 은닉층(hidden layer)을 포함하는 인공신경망입니다.
- 머신러닝에서는 비선형 분류를 위해 여러 trick을 사용하지만, DNN은 다수의 은닉층으로 비선형 분류가 가능해집니다.



# 심층신경망(Deep Neural Network, DNN)

## ■ input layer(입력층)

신경망의 첫 번째 레이어로서 입력 데이터를 수신합니다.

## ■ hidden layer(은닉층)

신경망에서 입력 레이어(특성)와 출력 레이어(예측) 사이에 위치하는 합성 레이어입니다.

신경망에 하나 이상의 히든 레이어가 포함될 수 있습니다.

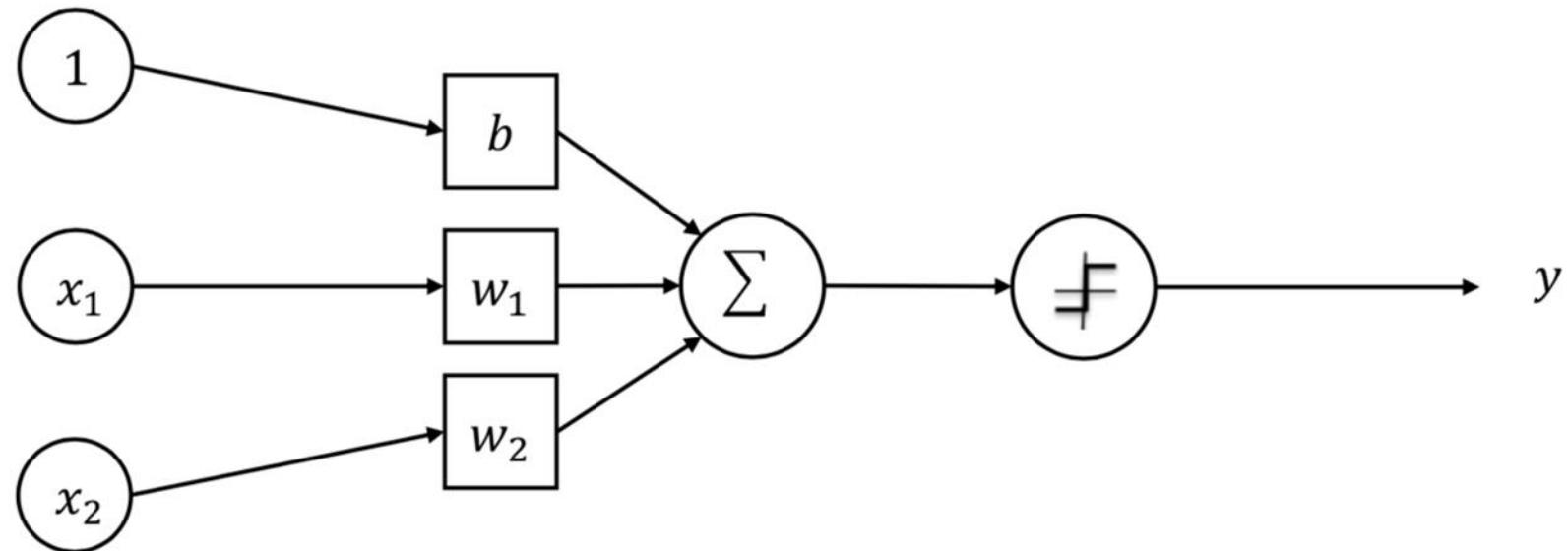
## ■ output layer(출력층)

신경망의 '최종' 레이어입니다. 이 레이어에 답이 포함됩니다.

## ■ Dense Layer(밀집층)

fully connected layer(완전 연결층)이나 dense layer(밀집층)라고 불리는 밀집 연결 층(densely connected layer)

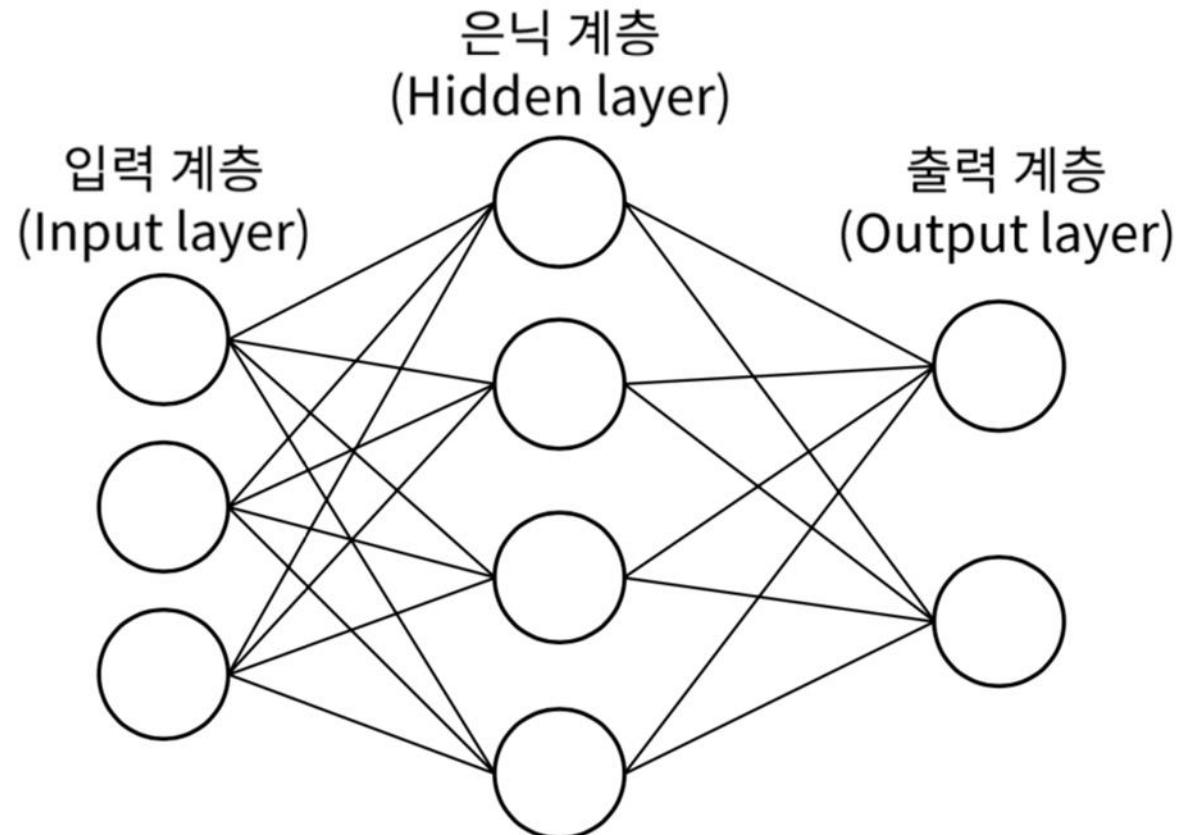
# 뉴런 Neuron



입력 (Input node)	가중치와 편향 (Weights and bias)	활성 함수 (Activation function)	출력 (Output node)
--------------------	-------------------------------	--------------------------------	---------------------

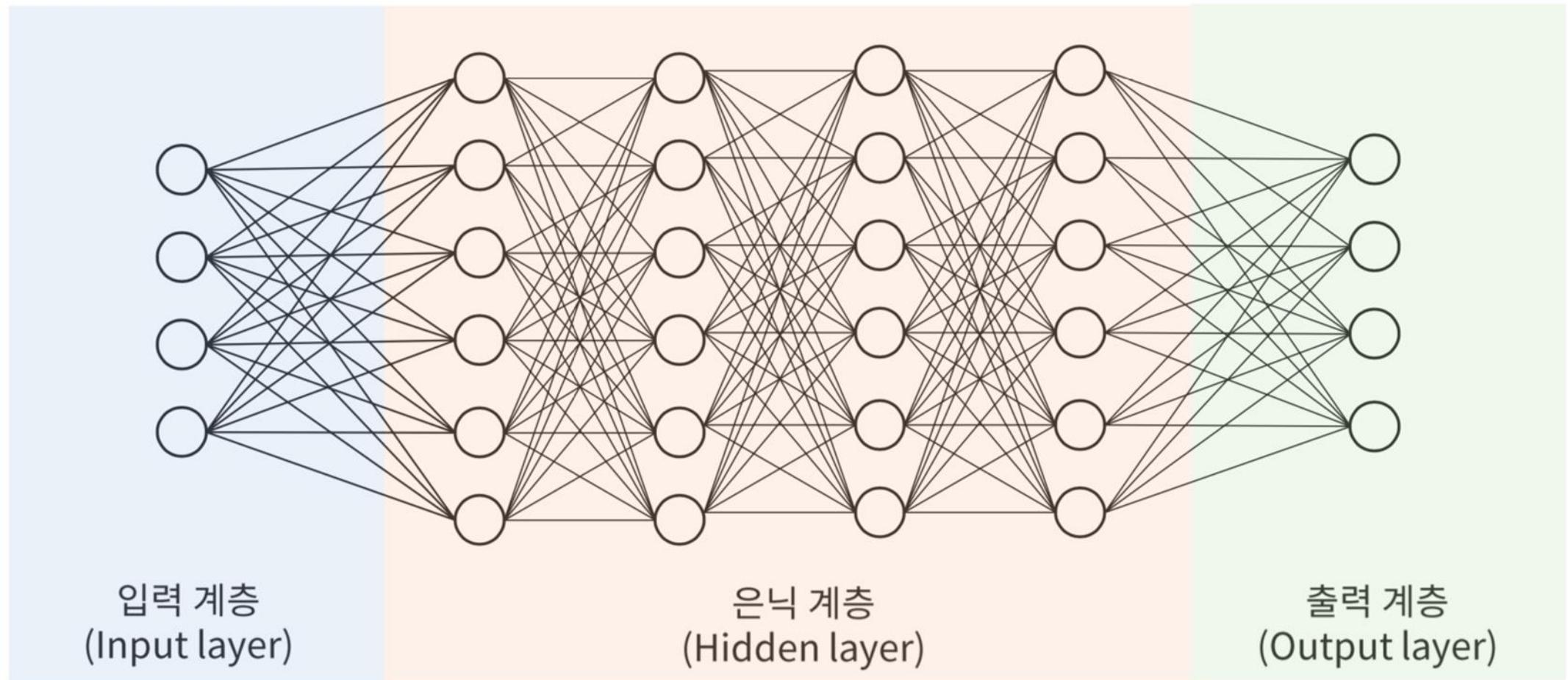
신경망은 **뉴런을 기본 단위**로 하며, 이를 조합하여 복잡한 구조를 이룬다.

# 얕은 신경망 Shallow Neural Network



가장 단순하고 얕은(은닉 계층이 1개인) 신경망 구조를 얕은 신경망이라고 한다.

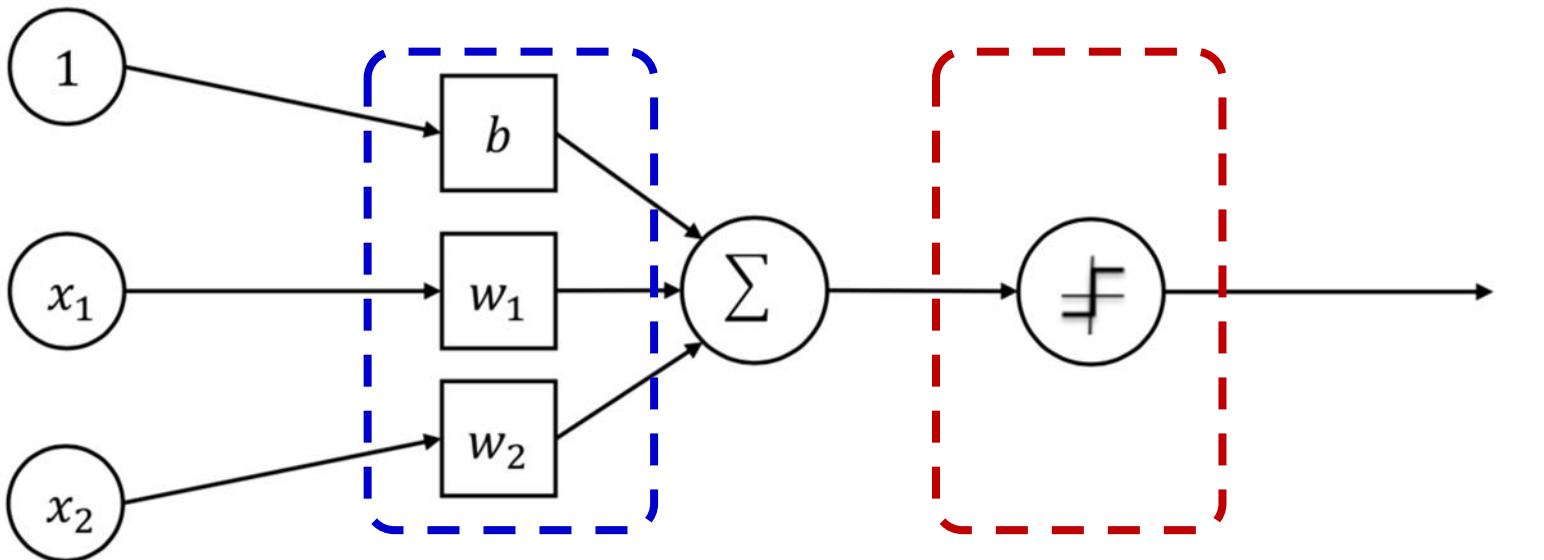
# 심층 신경망 Deep Neural Network (DNN)



- 얕은 신경망보다 은닉 계층이 많은 신경망을 DNN이라고 부른다.

# 가중치(Weight)

가중치는 입력값이 연산결과에 미치는 영향도를 조절하는 요소입니다.



입력  
(Input node)

가중치와 편향  
(Weights and bias)

활성 함수  
(Activation function)

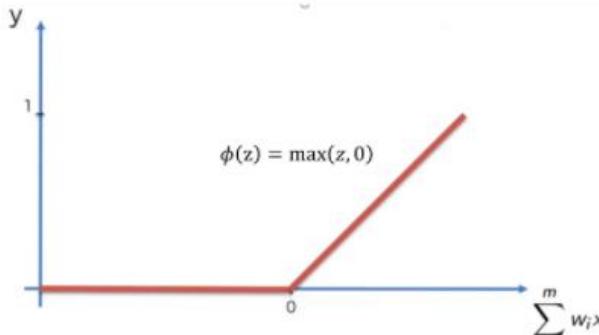
출력  
(Output node)

# 활성화 함수(Activation function)

입력값들의 수학적 선형결합을 다양한 형태의 비선형(또는 선형) 결합으로 변환하는 역할을 합니다.

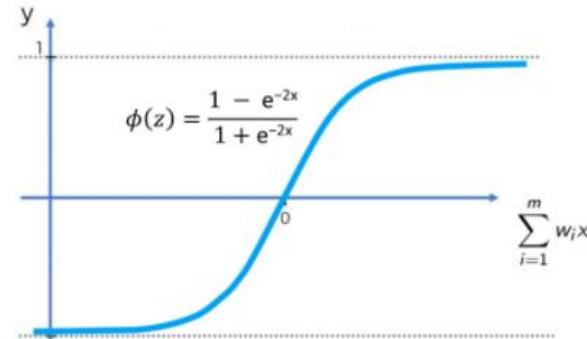
## 렐루 (ReLU)

입력이 양수일때는  $x$ , 음수일 때는 0을 출력



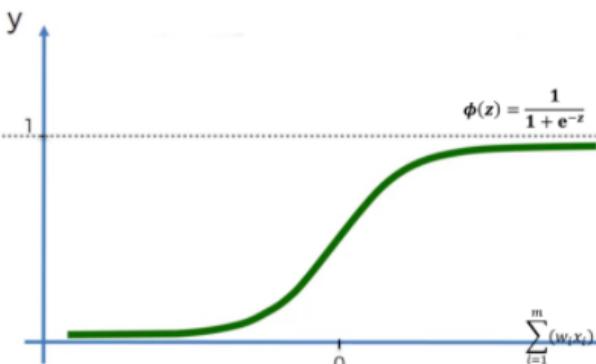
## 하이퍼볼릭 탄젠트 (Hyperbolic Tangent)

선형함수의 결과를 -1~1까지의 비선형 형태로 변경하는 함수



## 시그모이드 (Sigmoid)

0~1까지의 비선형 형태로 변경



## 소프트맥스 (Softmax)

입력값을 0~1 사이 출력이 되도록 정규화, 출력값들의 총합은 항상 1

$$\phi(z) = \frac{e^i}{\sum_{j=0}^k e^j} \quad | \text{where } i=0,1,\dots,k$$

# 손실함수(Loss Function)

인공신경망 학습의 목적함수로 출력값(예측값)과 정답(실제값)의 차이를 계산합니다.

## ■ 회귀예측(Regression)

평균제곱오차 : Mean Squared Error

평균절대오차 : Mean Absolute Error

## ■ 이진분류(Binary Classification)

Binary Cross Entropy

## ■ 다중분류( Binary Classification)

Categorical Cross Entropy : label(target, 출력값)이 원핫 벡터(One-Hot Vector)

Sparce Categorical Cross Entropy : label(target , 출력값)이 정수(0, 1, 2, 3 또는 n)

# 원핫벡터(One-Hot Vector, One-Hot Encoding)

고유 값에 해당하는 칼럼에만 1을 표시하고 나머지 칼럼에는 0을 표시하는 방법입니다.

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

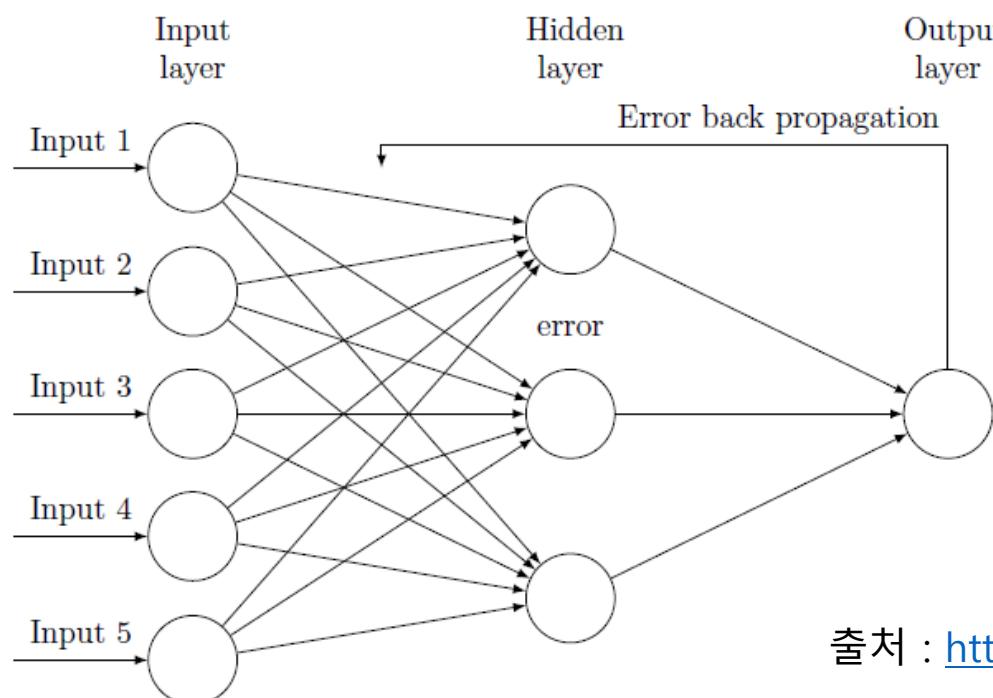
Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

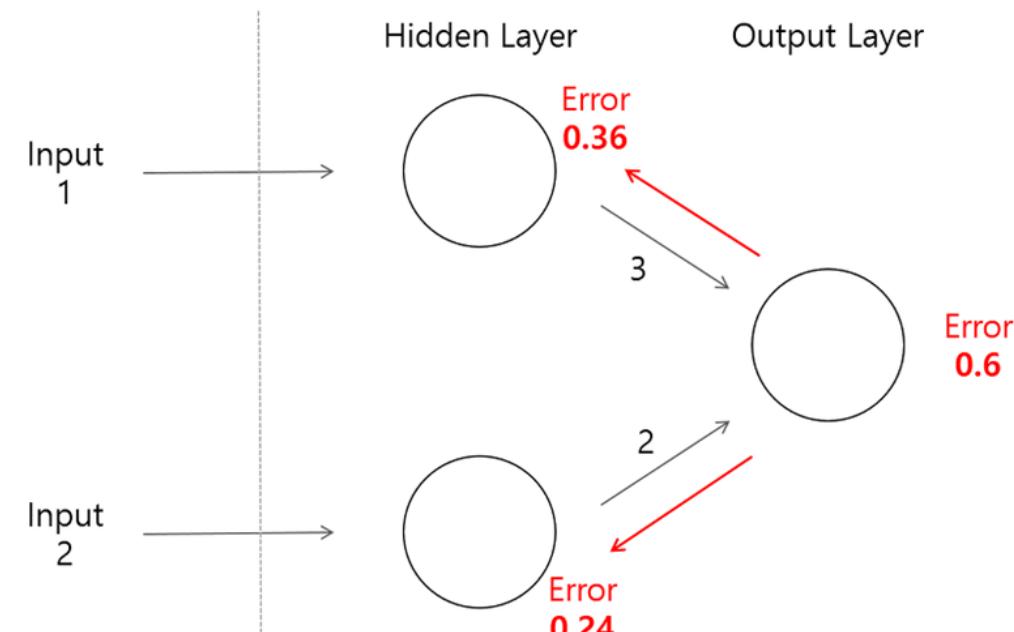


# 딥러닝 학습방법

- 딥러닝 학습은 손실/에러(Loss/Error)를 최소화 하는 인공신경망의 가중치( weight)와 편향(bias)을 찾는 과정입니다.
- 딥러닝 학습은 순전파(Forward Propagation)와 오차역전파(Error Back Propagation)의 반복으로 진행이 됩니다.
- 순전파는 뉴럴 네트워크의 입력층부터 출력층까지 순서대로 변수들을 계산하고 저장하는 것입니다.
- 오차역전파는 결과값을 통해서 역으로 input 방향으로 오차(Error)를 다시 보내며 가중치를 재업데이트 하는 것으로, 결과에 영향을 많이 미친 노드(뉴런)에 더 많은 오차를 돌려 줍니다.



출처 : <https://sacko.tistory.com/19>



# 딥러닝 학습방법

## ■ 딥러닝 학습방법

딥러닝 학습의 목표는 모델에 입력값을 넣었을 때의 출력값이 최대한 정답과 일치하게 하는 것입니다.

딥러닝 모델의 매개변수(weight, bias)를 무작위로 부여한 후,

반복학습(순전파-오차역전파)을 통해

모델의 출력값을 정답과 가깝게 되도록

매개변수(weight, bias)를 조금씩 조정합니다.

## ■ 순전파(Forward Propagation)

딥러닝 모델에 값을 입력해서 출력을 얻는 과정입니다.

## ■ 오차역전파(Error Backpropagation)

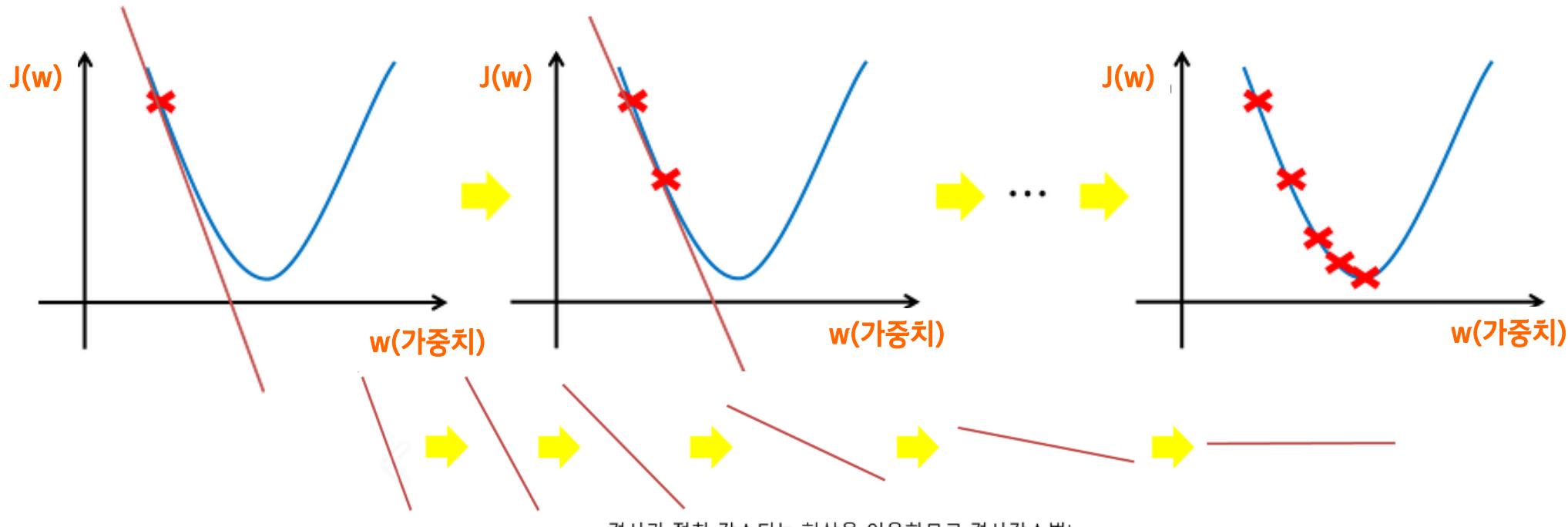
실제값과 모델 결과값에서 오차를 구해서, 오차를 입력(input) 방향으로 보내서 가중치를 재업데이트 하는 과정입니다.

## ■ 손실함수(Loss Function)

손실함수는 신경망 학습의 목적으로(목적함수) 모델의 출력값(예측값)과 정답(실제값)의 차이를 계산합니다.

# 경사하강법(Gradient Descent)

- 손실함수  $J(w)$ 는 가중치( $w$ )의 함수로, 볼록함수 형태라면 미분으로 손실이 가장 작은 가중치를 찾을 수 있습니다.
- 하지만, 딥러닝에서는 손실함수가 복잡하고 계산량이 매우 크고, 미분이 0이 되는 값이 여러 개 존재하므로 미분만으로 최소값을 찾기 어려워 경사하강법(Gradient Descent)을 사용합니다.
- 경사하강법은 손실함수의 현 가중치에서 기울기를 구해서 손실(Loss)을 줄이는 방향으로 업데이트 해 나갑니다.

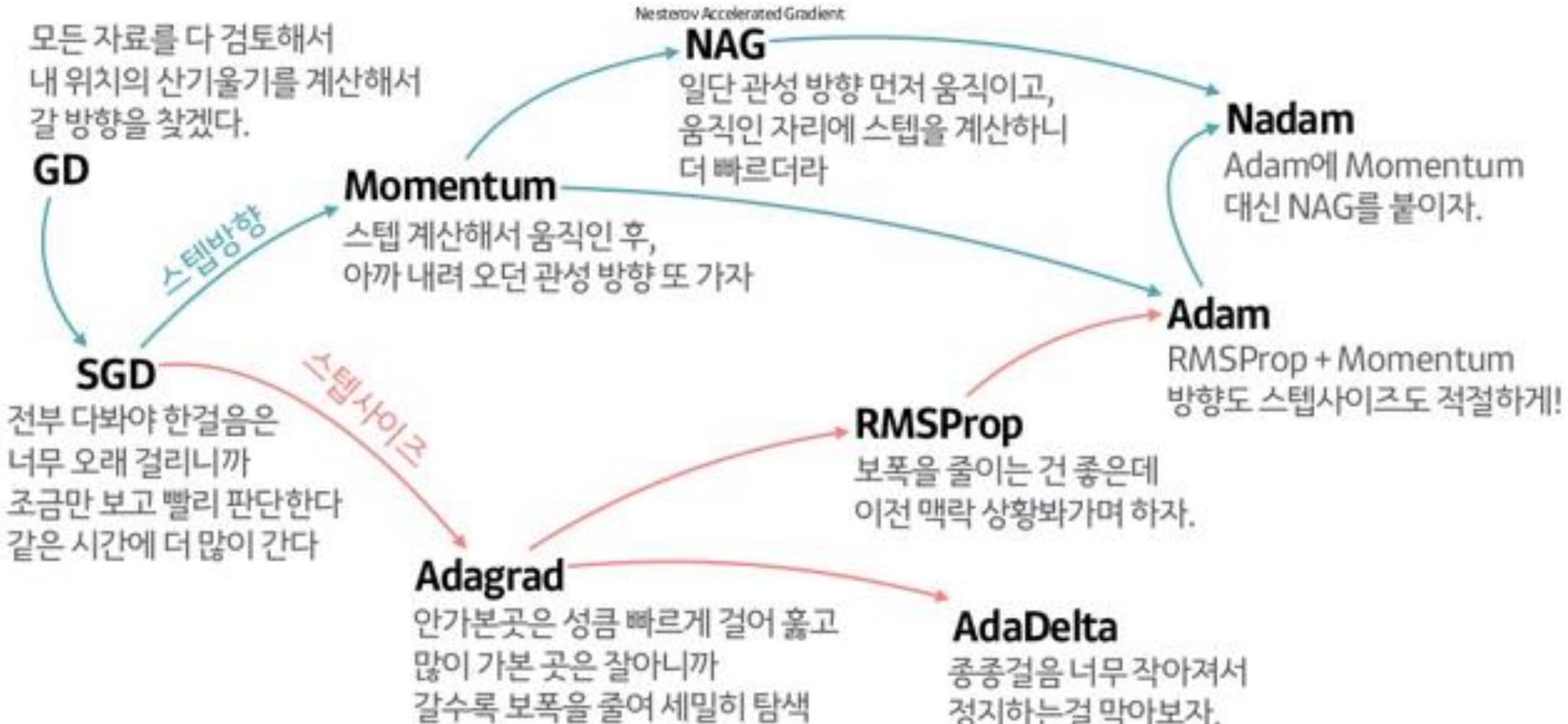


경사가 점차 감소되는 현상을 이용하므로 경사감소법!

참고 : [https://angeloyeo.github.io/2020/08/16/gradient\\_descent.html](https://angeloyeo.github.io/2020/08/16/gradient_descent.html)  
[https://angeloyeo.github.io/2020/08/16/gradient\\_descent.html](https://angeloyeo.github.io/2020/08/16/gradient_descent.html)

# 옵티마이저(Optimization Algorithm)

손실함수를 최소화하는 방향으로 가중치를 갱신하는 알고리즘입니다.



# 옵티마이저(Optimization Algorithm)

## ■ 최적화(Optimization)

딥러닝 모델의 매개변수(weight, bias)를 조절해서 손실함수의 값을 최저로 만드는 과정으로 경사하강법(Gradient Descent)이 대표적입니다.

## ■ 경사하강법(gradient descent)

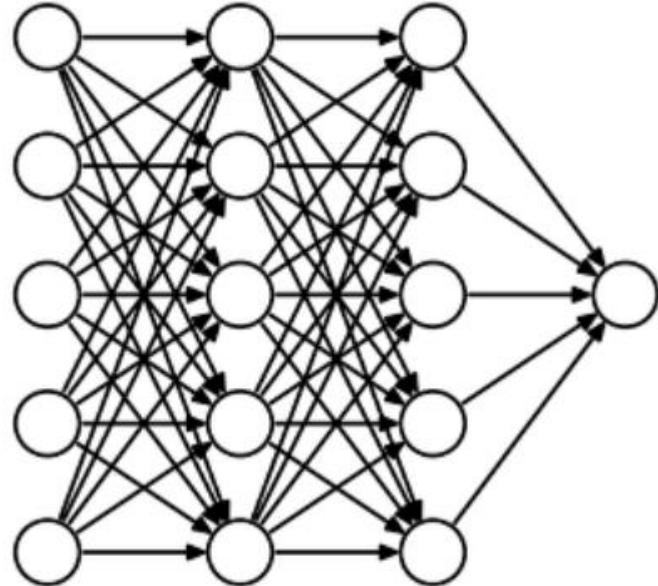
학습 데이터의 조건에 따라 모델의 매개변수를 기준으로 손실의 경사를 계산하여 손실을 최소화하는 기법입니다. 쉽게 설명하면, 경사하강법은 매개변수를 반복적으로 조정하면서 손실을 최소화하는 가중치와 편향의 가장 적절한 조합을 점진적으로 찾는 방식입니다.

## ■ 경사(gradients)

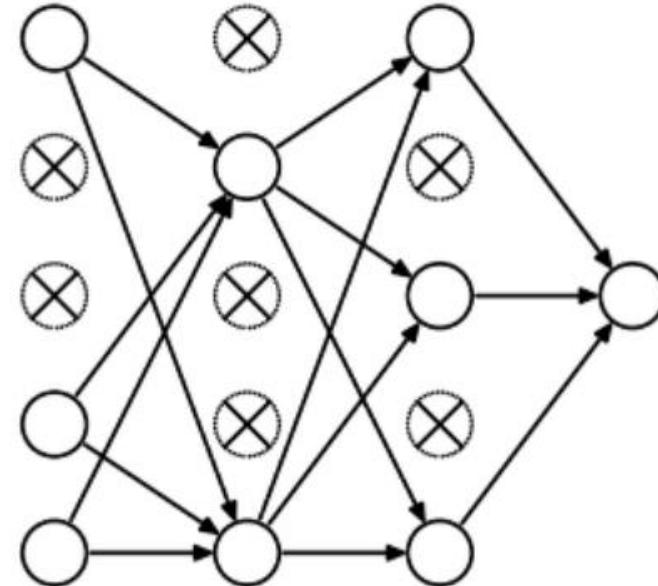
모든 독립 변수를 기준으로 한 편미분의 벡터입니다. 머신러닝에서 경사는 모델 함수의 편미분의 벡터입니다.

# 드롭아웃(Dropout)

Hidden Layer의 일부 유닛이 동작하지 않게 하여 overfitting(과적합)을 막는 방법입니다.



일반 심층신경망



Dropout이 적용된 심층신경망

## ■ 과적합(overfitting)

생성된 모델이 학습 데이터와 지나치게 일치하여 새 데이터를 올바르게 예측하지 못하는 경우입니다.

## ■ 일반화(generalization)

모델학습에 사용된 데이터가 아닌 이전에 접하지 못한 새로운 데이터에 대해 올바른 예측을 수행하는 능력을 의미합니다.

# 미니배치(Mini Batch)

데이터가 수십만개 이상이라면 전체 데이터를 더 작은 단위로 나누어서 학습하는 방법입니다.



1 Epoch(에폭) : 전체 데이터셋을 한번 학습

1 iteration(이터레이션) : 1회 학습

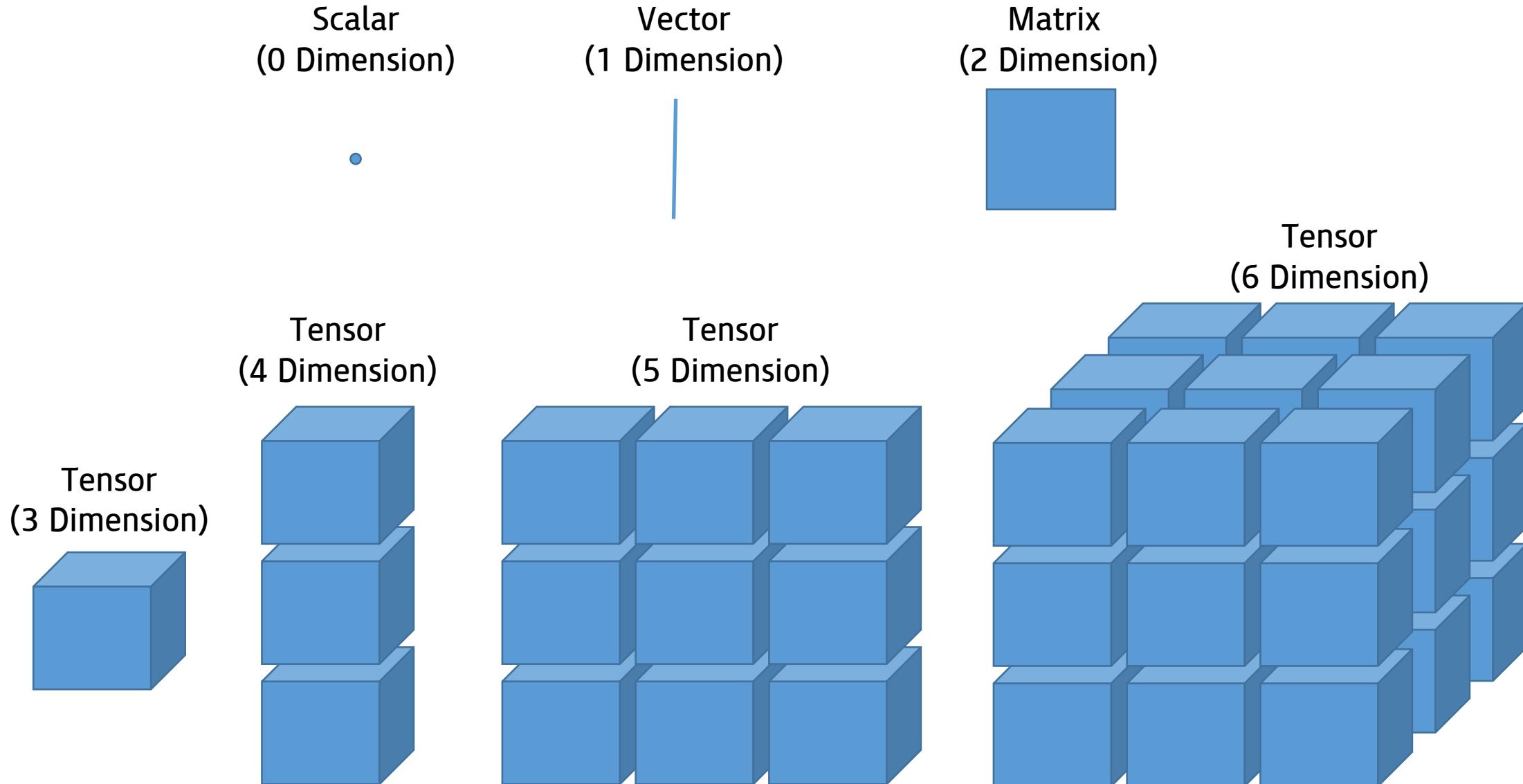
Minibatch : 데이터셋을 batch size 크기로 나누어서 학습

ex) 총 데이터가 1,000,000개, batch size가 1,000 이면,

1 iteration = 1,000개 데이터에 대해서 학습

1 Epoch =  $1,000,000/\text{batch size} = 1,000 \text{ iteration}$

# 텐서(Tensor)



# TensorFlow

딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 프레임워크입니다.

<https://www.tensorflow.org/tutorials/>

The screenshot shows the TensorFlow website's 'TensorFlow 시작하기' (Getting Started) page. On the left, there is a sidebar with various navigation links such as 'ML 학습 및 사용', '연구 및 실험', '프로덕션 규모의 ML', etc. The main content area features a large heading 'TensorFlow 시작하기'. Below it, a text block explains that TensorFlow is a research and production-ready open-source machine learning library. It provides a link to the 'TensorFlow Keras 가이드' for getting started. To the right, there is a code editor window displaying Python code for a Keras model to classify MNIST digits. At the bottom of the code editor, there is a blue button labeled 'Keras 가이드 읽기'.

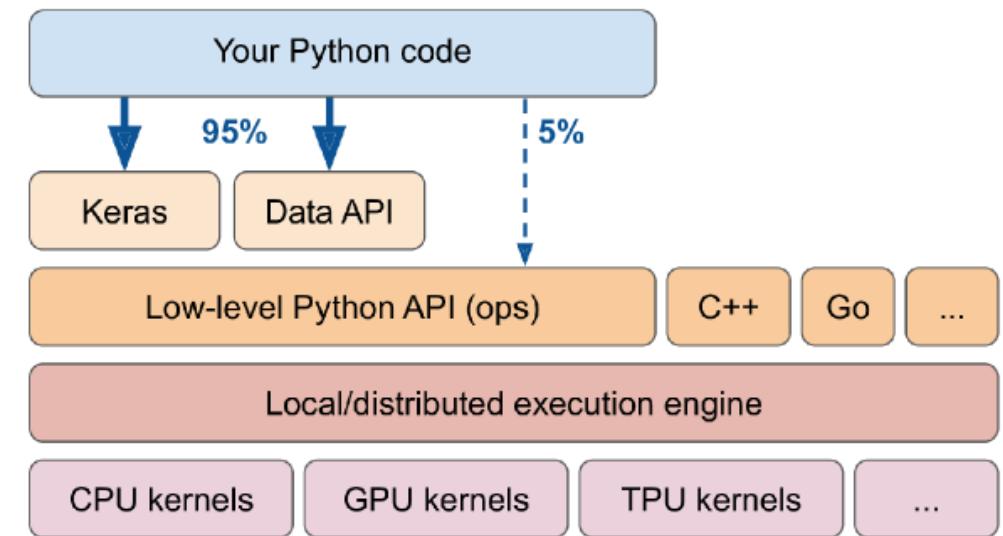
```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

텐서플로 구조



# TF2 모델 구성 방법

## 1. Sequential Model

Sequential Model은 직관적이며 간결하여 딥러닝을 처음 접하는 사용자에게 적합합니다.

### ■ 방법 1

```
# Passing a list of layers to the constructor
model = Sequential([
    Dense(5, activation='relu', input_shape=(4,)),
    Dense(10, activation='relu'),
    Dense(3, activation='softmax'),
])
```

### ■ 방법 2

```
# Adding layer via add() method
model = Sequential()
model.add(Dense(5, activation='relu', input_shape=(4,)))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

# TF2 모델 구성 방법

## 2. Functional API

입력과 다중 출력 등 복잡한 모델을 정의할 수 있습니다.

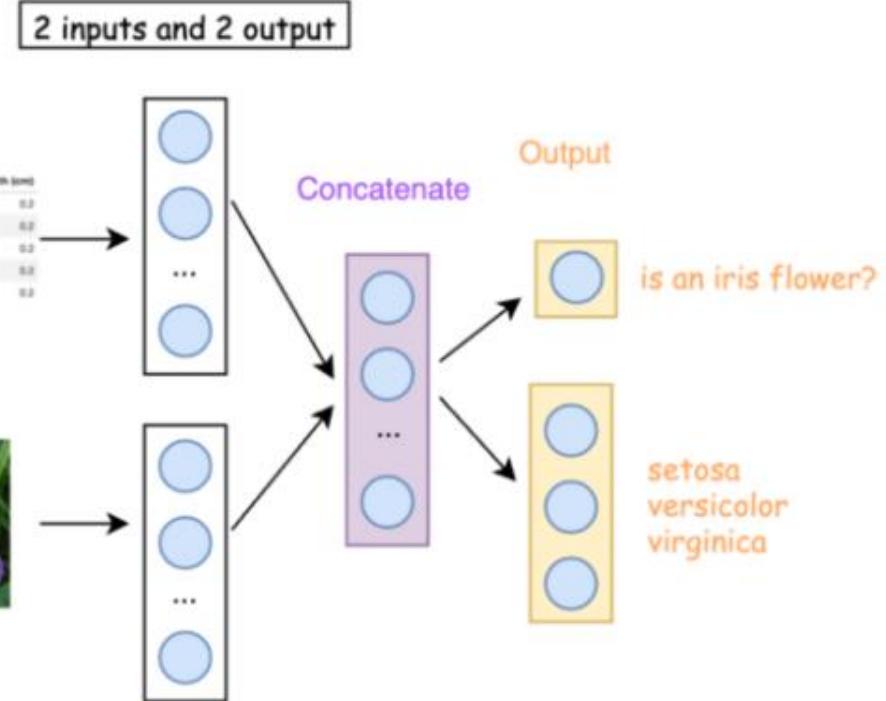
```
# This returns a tensor
inputs = Input(shape=(4,))

# A layer instance is callable on a tensor, e
x = Dense(5, activation='relu')(inputs)
x = Dense(10, activation='relu')(x)
outputs = Dense(3, activation='softmax')(x)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=outputs)
```

2 inputs and 2 output

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2



# TF2 모델 구성 방법

## 3. Model Subclassing

가장 자유롭게 모델을 구축할 수 있는 방법입니다.

```
class CustomModel(Model):

    def __init__(self, **kwargs):
        super(CustomModel, self).__init__(**kwargs)
        self.dense1 = Dense(5, activation='relu', )
        self.dense2 = Dense(10, activation='relu')
        self.dense3 = Dense(3, activation='softmax')

    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        return self.dense3(x)

my_custom_model = CustomModel(name='my_custom_model')
```

# AI 모델 구현



10-DNN.ipynb

- ① 라이브러리 임포트(import)
- ② 데이터 가져오기>Loading the data)
- ③ 탐색적 데이터 분석(Exploratory Data Analysis)
- ④ 데이터 전처리(Data PreProcessing) : 데이터타입 변환, Null 데이터 처리, 누락데이터 처리, 카테고리 데이터, 더미특성 생성, 특성 추출 (feature engineering) 등
- ⑤ 훈련/테스트 데이터 분할(Train Test Split)
- ⑥ 데이터 정규화(Normalizing the Data)
- ⑦ 모델 개발(Creating the Model)
- ⑧ 모델 성능 평가(Evaluating Model Performance)

# AI 모델 학습 데이터



데이터파일 : churn\_data.csv

customerID	gender	SeniorCitizen	TotalCharges	Churn
7590-VHVEG	Female	0	29.85	No
5575-GNVDE	Male	0	1889.5	No
3668-QPYBK	Male	0	108.15	Yes
7795-CFOCW	Male	0	1840.75	No
9237-HQITU	Female	0	151.65	Yes
9305-CDSKC	Female	0	820.5	Yes
1452-KIOVK	Male	0	1949.4	No
6713-OKOMC	Female	0	301.9	No

- customerID: 고객ID
- gender: 고객 성별
- SeniorCitizen: 고객이 노약자인가 아닌가
- Partner: 고객에게 파트너가 있는지 여부(결혼 여부)
- Dependents: 고객의 부양 가족 여부
- tenure: 고객이 회사에 머물렀던 개월 수
- PhoneService: 고객에게 전화 서비스가 있는지 여부
- MultipleLines: 고객이 여러 회선을 사용하는지 여부
- InternetService: 고객의 인터넷 서비스 제공업체
- OnlineSecurity: 고객의 온라인 보안 여부
- OnlineBackup: 고객이 온라인 백업을 했는지 여부
- DeviceProtection: 고객에게 기기 보호 기능이 있는지 여부
- TechSupport: 고객이 기술 지원을 받았는지 여부
- StreamingTV: 고객이 스트리밍TV를 가지고 있는지 여부
- StreamingMovies: 고객이 영화를 스트리밍하는지 여부
- Contract: 고객의 계약기간
- PaperlessBilling: 고객의 종이 없는 청구서 수신 여부(모바일 청구서)
- PaymentMethod: 고객의 결제 수단
- MonthlyCharges: 매월 고객에게 청구되는 금액
- TotalCharges: 고객에게 청구된 총 금액
- Churn: 고객 이탈 여부 (label, y)

# 심층신경망 구현 - 라이브러리, 데이터

## ■ 라이브러리 임포트

```
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense, Activation, Dropout
```

## ■ 데이터 로드

```
df = pd.read_csv('churn_data.csv')
```

# 심층신경망 구현 - 데이터 분석, 전처리

## ■ 데이터 분석

```
df.info()  
df.isnull().sum()  
df.describe().transpose()  
df.corr()['MonthlyCharges'][:-1].sort_values().plot(kind='bar')  
sns.pairplot(df)
```

## ■ 데이터 전처리

```
df.drop('customerID', axis=1, inplace=True)  
df['TotalCharges'].replace([' '], ['0'], inplace=True)  
df['TotalCharges'] = df['TotalCharges'].astype(float)  
df['Churn'].replace(['Yes', 'No'], [1, 0], inplace=True)
```

# 심층신경망 구현 - 데이터 전처리

## ■ 데이터 전처리

```
cols = ['gender', 'Partner', 'Dependents', 'PhoneService',
'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
'Contract', 'PaperlessBilling', 'PaymentMethod']

dummies = pd.get_dummies(df[cols], drop_first=True)
df = df.drop(cols, axis=1)
df = pd.concat([df, dummies], axis=1)

# df = pd.get_dummies(df)
# cols = list(df.select_dtypes('object').columns)
```

# 심층신경망 구현 - 데이터셋 분할

## ■ 훈련/테스트 데이터셋 분할

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop('Churn', axis=1).values
```

```
y = df['Churn'].values
```



```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y,  
    test_size=0.3,  
    random_state=42)
```

```
X_train.shape
```

```
[Out] (4930, 30)
```

```
y_train.shape
```

```
[Out] (4930,)
```

# 심층신경망 구현 - 데이터셋 정규화

## ■ 데이터셋 정규화/스케일링(Normalizing/Scaling)

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()  
scaler.fit(X_train)  
X_train = scaler.transform(X_train)  
X_test = scaler.transform(X_test)
```



# 심층신경망 구현 - 모델 구성

## ■ 모델 구성

```
model = Sequential()
```

*# input Layer*

```
model.add(Dense(64, activation='relu', input_shape=(30,)))
```

*# hidden Layer*

```
model.add(Dense(64, activation='relu'))
```

*# hidden Layer*

```
model.add(Dense(32, activation='relu'))
```

*# output Layer*

```
model.add(Dense(1, activation='sigmoid'))
```

## 심층신경망 구현 - 모델 구성(과적합 방지)



```
model = Sequential()
# input Layer
model.add(Dense(128, activation='relu', input_shape=(30,)))

# hidden Layer
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))

# output Layer
model.add(Dense(1, activation='sigmoid'))
```

# 심층신경망 구현 - 출력층 Activation Function

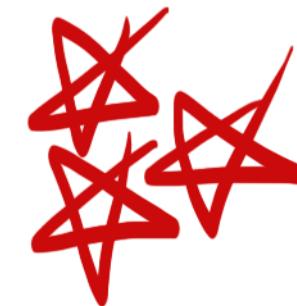
## ■ 회귀예측

```
model.add(Dense(1))
```

## ■ 이진분류

```
model.add(Dense(1, activation='sigmoid'))
```

sigmoid : 선형함수의 결과를 0~1까지의 비선형 형태로 변형하기 위한 함수



## ■ 다중분류

```
model.add(Dense(3, activation='softmax'))
```

softmax : 입력받은 값을 0~1 사이 출력이 되도록 정규화하여 출력값들의 총합이 항상 1 되는 특성을 가진 함수

# 심층신경망 구현 - 모델 컴파일

## ■ 회귀예측

```
model.compile(optimizer='adam', loss='mse')
```

## ■ 이진분류

```
model.compile(optimizer='adam', loss='binary_crossentropy',  
              metrics=['accuracy'])
```



## ■ 다중분류

```
model.compile(optimizer='adam', loss='categorical_crossentropy',  
              metrics=['accuracy'])
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])
```

# 심층신경망 구현 - 모델학습

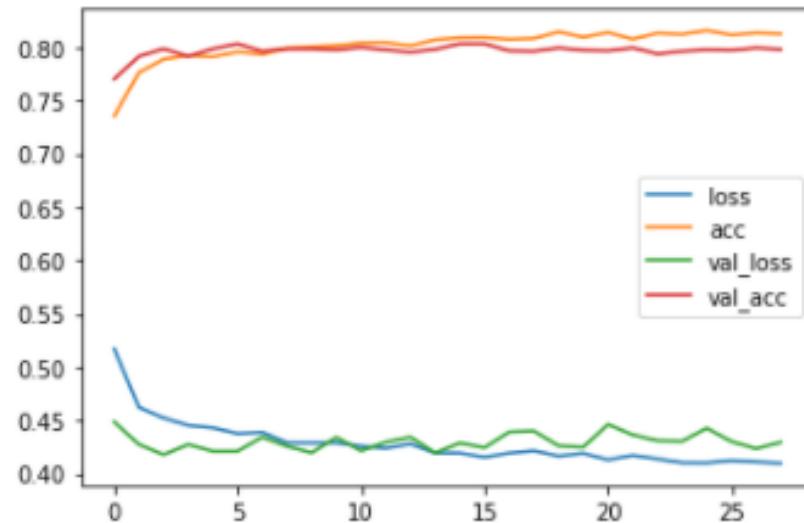
## ■ 모델 학습(훈련)

```
model.fit(X_train, y_train,  
          validation_data=(X_test, y_test), epochs=20, batch_size=10)
```



## ■ 학습과정 시각화

```
losses = pd.DataFrame(model.history.history)  
losses[['loss','val_loss']].plot()
```



# 심층신경망 구현 실습



11-DNN-Exercise.ipynb

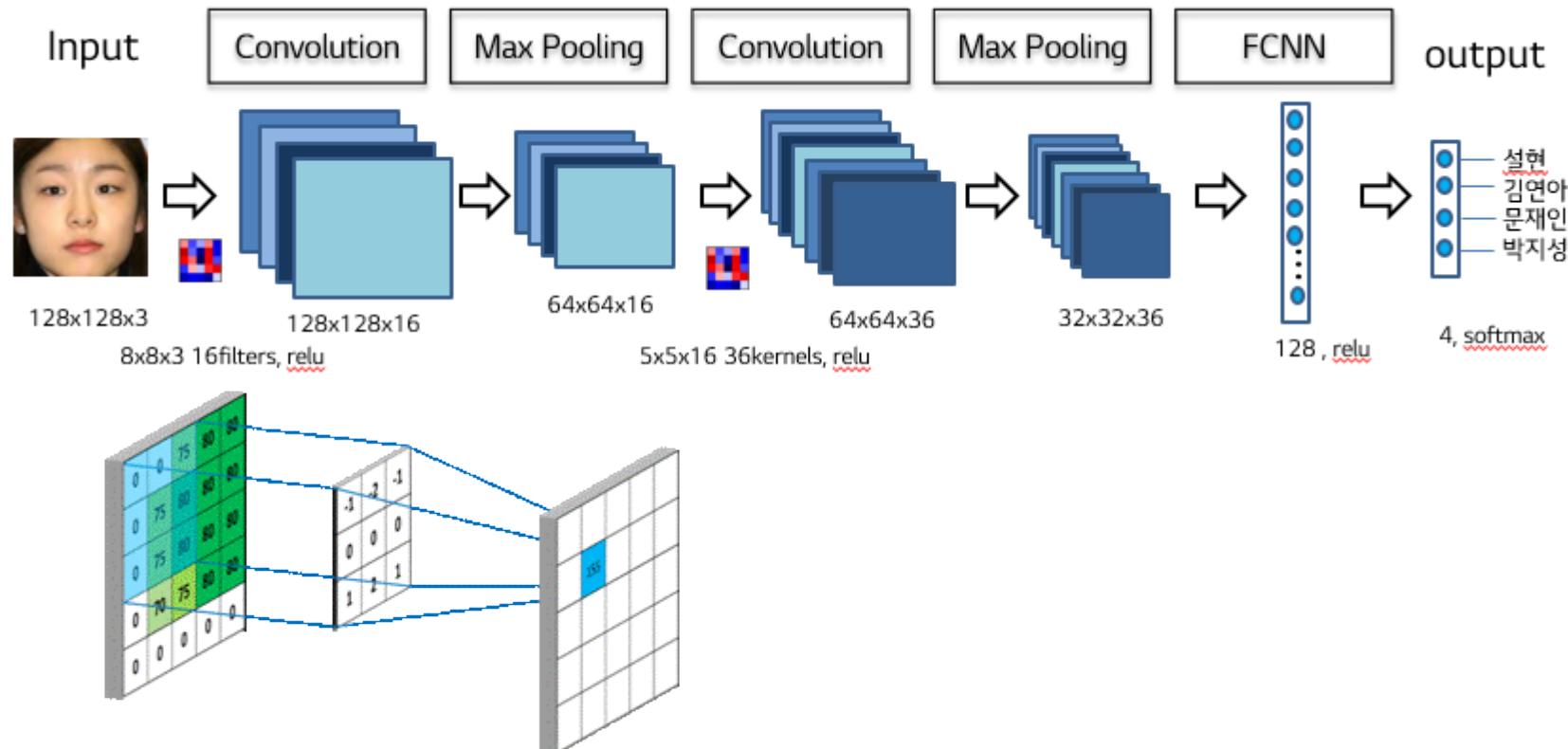


1. `X_train, X_valid` 값을 1,0 사이의 값으로 스케일링 하세요.
2. 딥러닝 심층신경망모델(DNN)로 통신사이탈고객을 예측하는 분류기를 만드세요.  
검증정확도(`val_acc`)가 80% 이상이 나오도록 하이퍼 파라미터를 설정하세요.  
그리고, 검증 정확도가 가장 높은 모델을 파일명 `best_model.h5`로 저장하세요.
3. 학습 정확도, 학습손실, 검증 정확도, 검증손실을 그래프로 표시하세요.

# 6. Further Study

# 합성곱 신경망(CNN, Convolutional Neural Network)

사람의 시각 피질 메커니즘에 영감을 받아 설계된 이미지, 영상등을 인식하는 신경망 모델  
Convolution층에서는 각 filter가 입력 이미지의 픽셀 전체를 차례로 훑고 지나가며  
linear combination을 진행하고 Feature Map을 구성합니다.

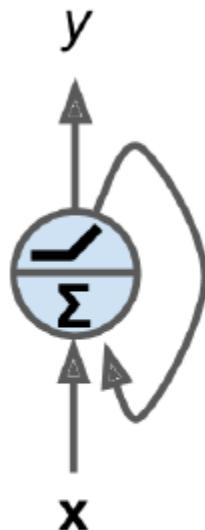


# 순환 신경망(RNN, Recurrent Neural Network)

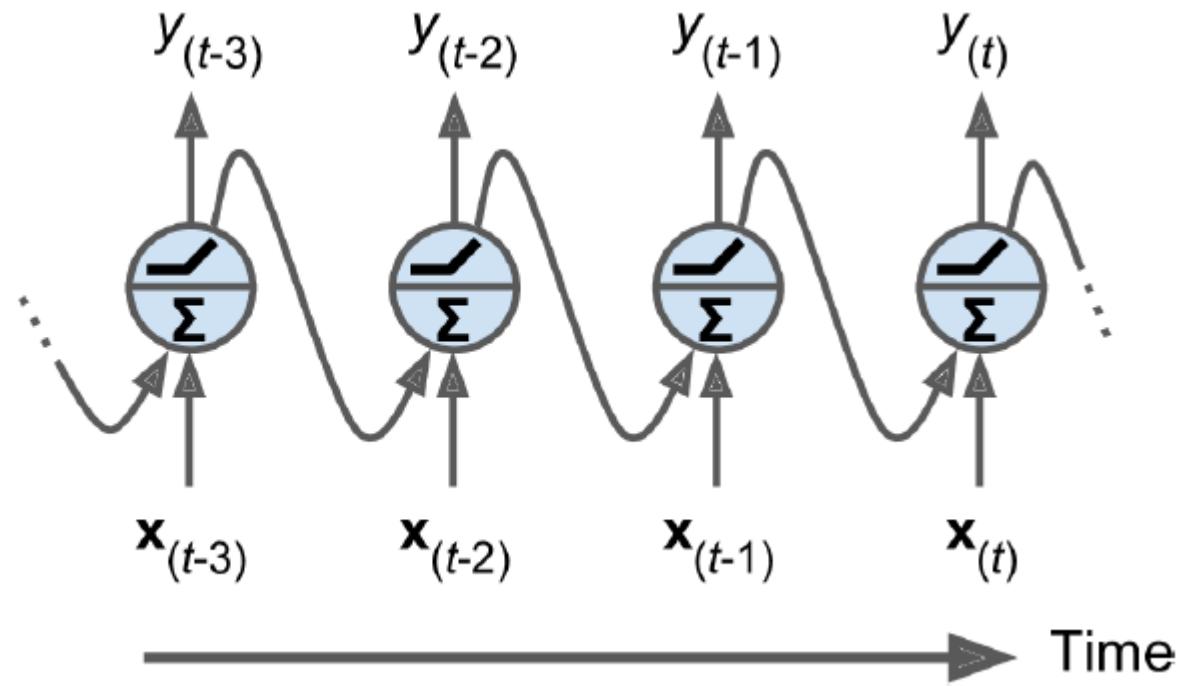
순환신경망은 고정 길이 입력이 아닌 임의 길이를 가진 시퀀스를 다룰 수 있습니다.

순환신경망은 시계열 데이터를 분석해서 미래값을 예측하고 문장, 오디오를 입력으로 받아 자동번역, 자연어처리에 유용합니다.

## ■ 순환뉴런

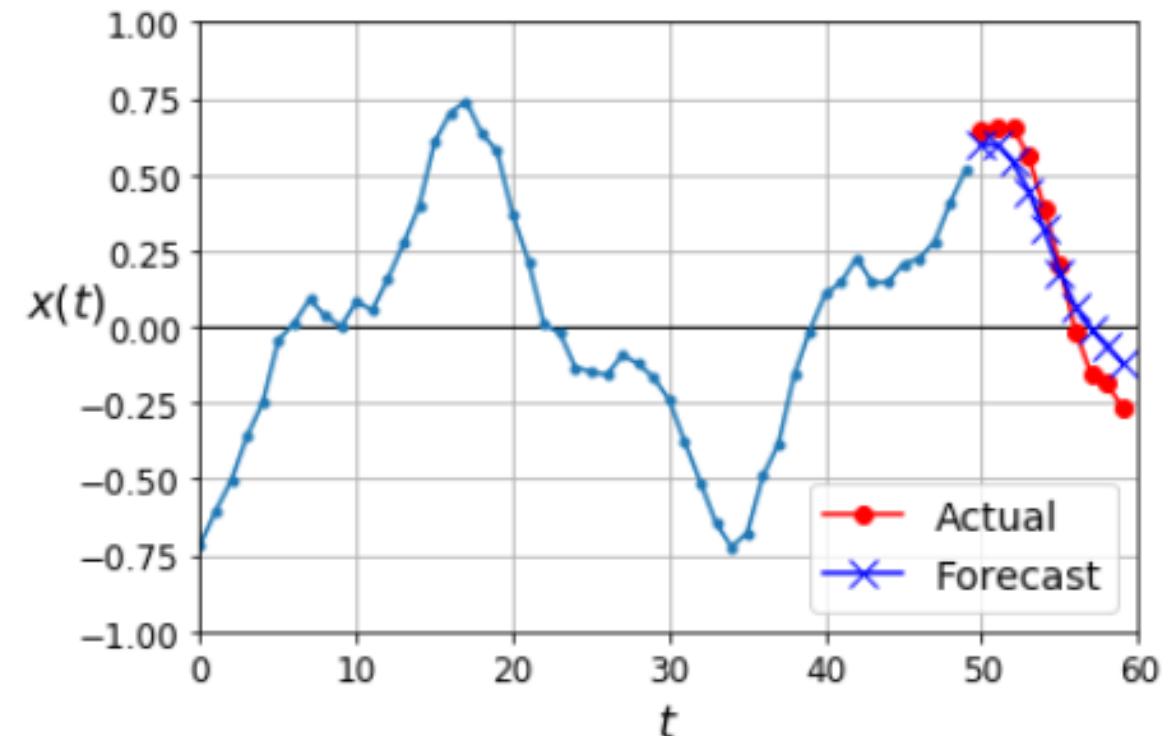
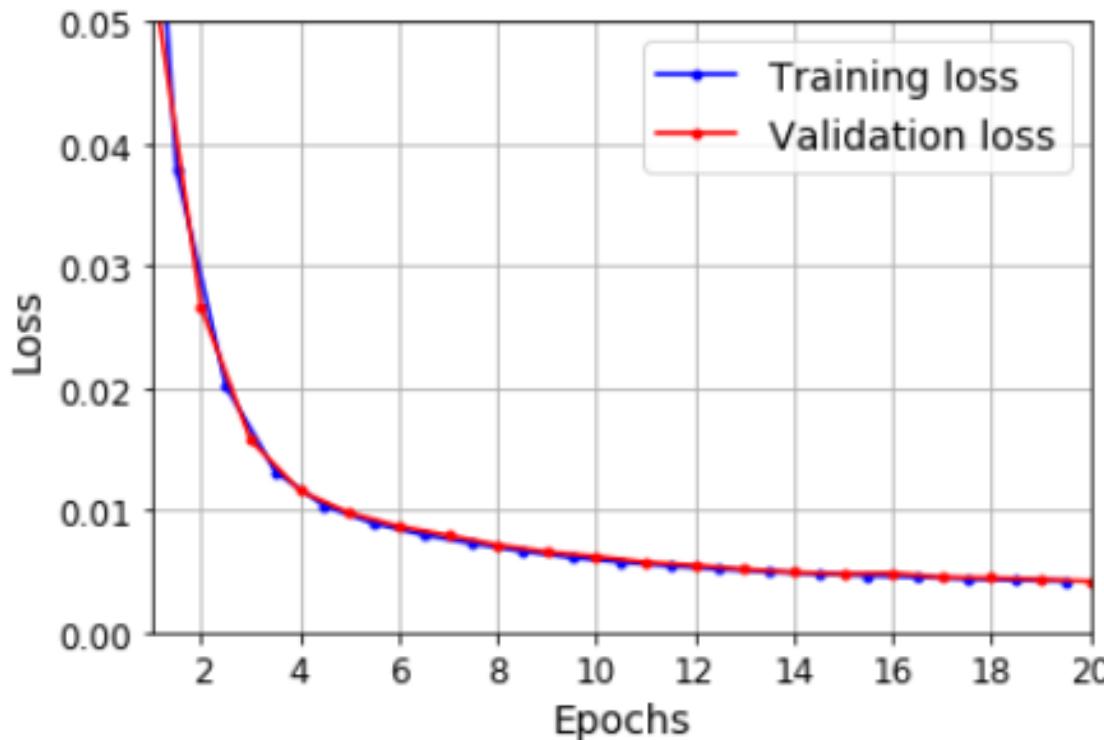


## ■ 순환뉴런을 타임 스텝으로 펼친 모습



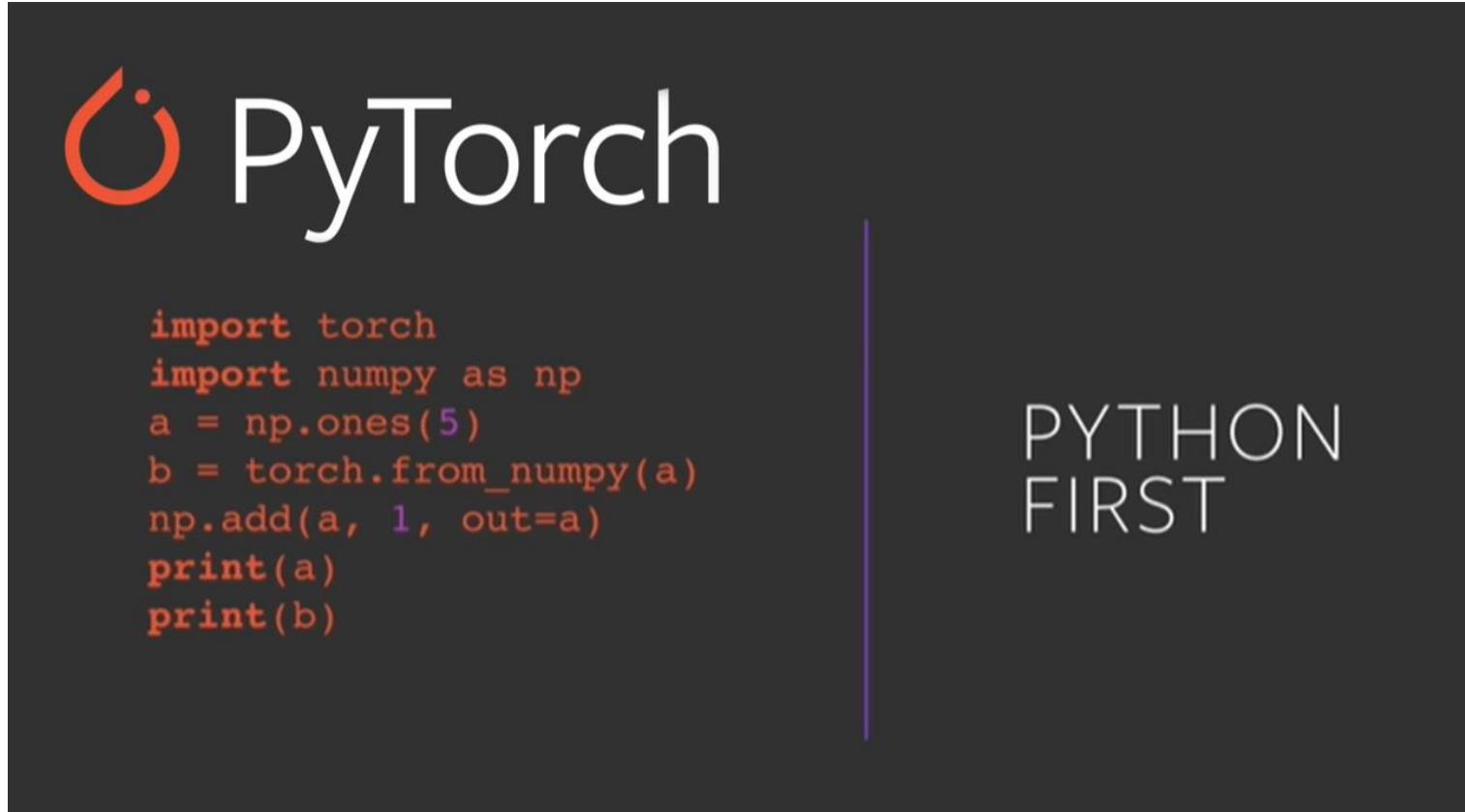
# 시퀀스 처리

[https://github.com/rickiepark/handson-ml2/blob/master/15\\_processing\\_sequences\\_using\\_rnns\\_and\\_cnns.ipynb](https://github.com/rickiepark/handson-ml2/blob/master/15_processing_sequences_using_rnns_and_cnns.ipynb)



# 파이토치(PyTorch)

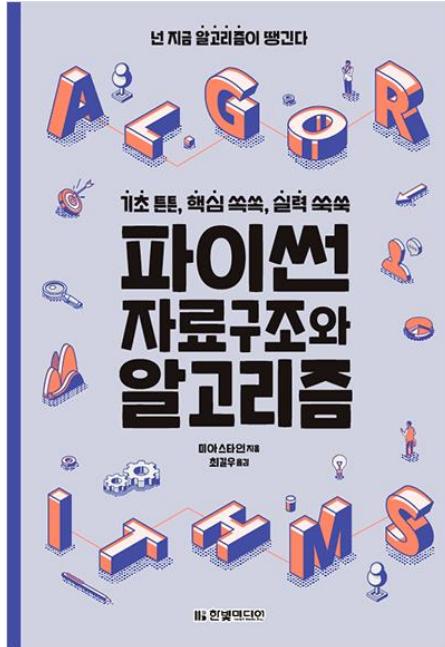
파이토치(PyTorch) 튜토리얼 <https://tutorials.pytorch.kr/>



PyTorch로 시작하는 딥러닝 입문 <https://wikidocs.net/book/2788>

# 추천 도서

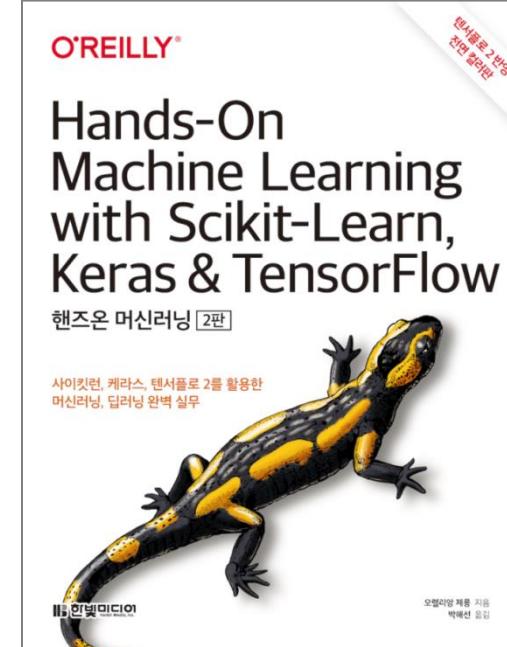
<https://bit.ly/38cErVX>



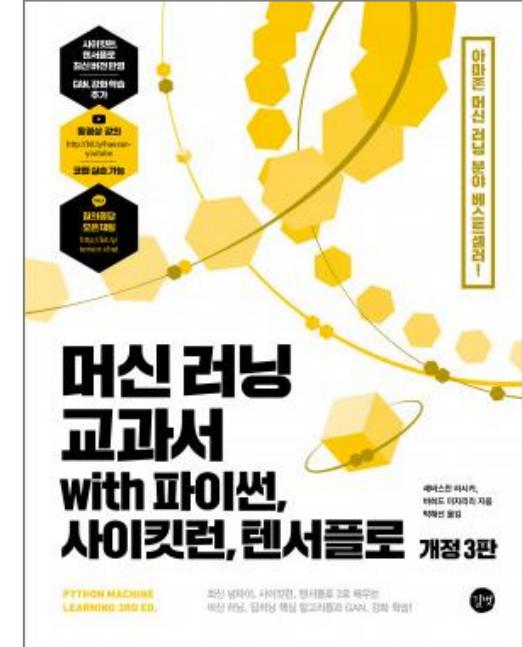
<https://bit.ly/3tbyWQf>



<https://bit.ly/36Ltr2X>



<https://bit.ly/3rRjiZn>



<https://wikidocs.net/>  
WikiDocs





수고하셨습니다. 감사합니다.