

# 파이썬 기반 데이터 사이언스 입문

박정희 과장

디지털혁신실 디지털신기술팀

2024년 11월 28일 - 29일

# 데이터 분석

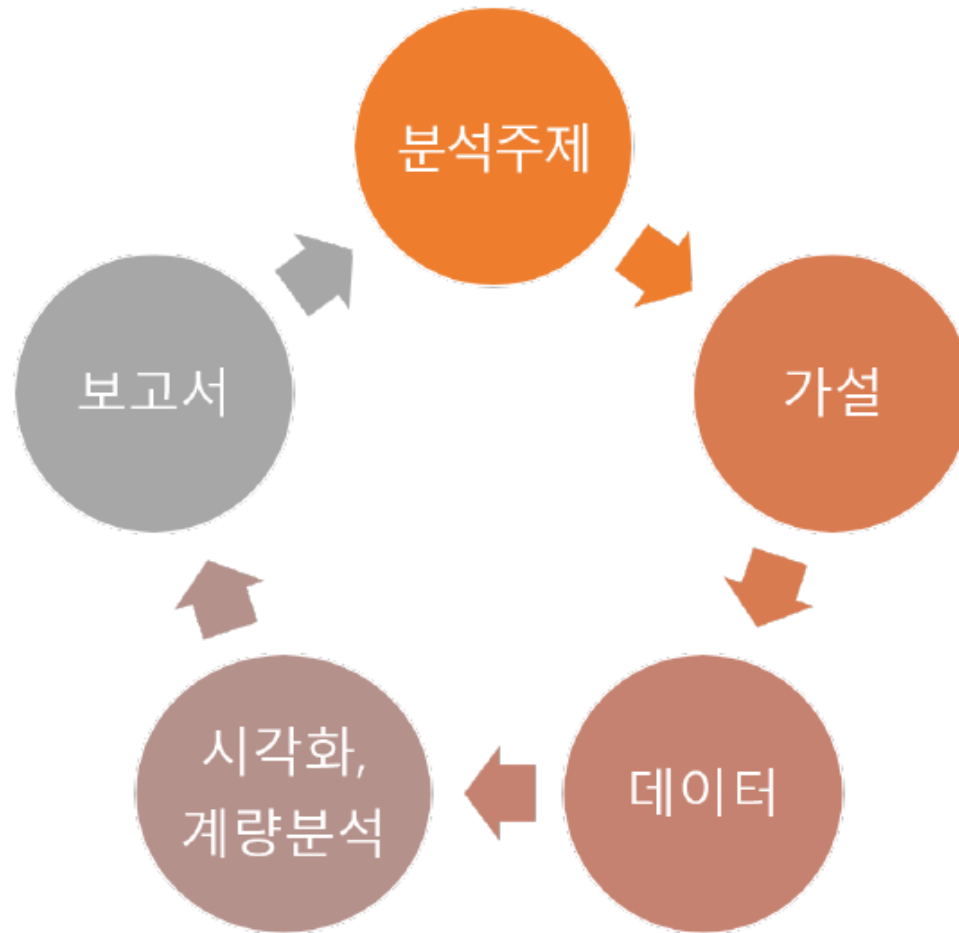
# 단계별 데이터 분석

---

- 분석주제(질문)
- 가설(변수 간 관계에 대한 이론 · 모형)
- 데이터 준비(입수, 전처리, 가공)
- 분석(통계, 상관/회귀계수, 전망, 머신러닝)
- 시각화(선, 막대기, 히트맵/네트워크)
- 보고서(내러티브, 기승전결)

# 애자일(agile) 프로세스

---



# 애자일(agile) 프로세스

---

- 애자일 프로세스의 가장 큰 걸림돌(병목 지점)은 데이터 준비 과정  
(국민계정)



- 분석주제·가설, 분석방법론 등은 주로 본격적인 프로세스 시작 전 완료  
(부서별 수요(정책·조사연구, 모니터링 등), 데이터, 기존 연구 및 학술지식 등에 의해 결정)
- 입수 ·전처리 ·가공 및 분석 ·시각화를 위한 두 가지 방법: Excel, Pipeline

# Excel

---

- 엑셀에서의 데이터 작업은 손쉽고 직관적
  1. 누구나 엑셀 프로그램을 열고 숫자를 입력하거나 읽을 수 있음
  2. 현재 작업 중인 데이터를 눈으로 확인
- 데이터가 커지고 분석 프로세스가 길어지면서 처리 시간이 급격하게 증가
  1. 아무리 모니터가 커도 데이터가 한 눈에 안들어옴
  2. 여러 시트에 산재된 데이터가 어떻게 생성되었는지, 오류는 없는지 확인하기 어려움

# Pipeline

---

- 데이터 분석 파이프라인(pipeline)은
  1. 데이터 속성에 맞도록 데이터를 구조화하고
  2. 독립적 기능을 수행하는 하위 프로세스들로 구성되어 데이터 준비, 분석·시각화를 효율적으로 수행
- 정확한 연구주제(질문)에 대한 강건한(robust) 분석결과(답)를 얻기 위해서는 다양한 데이터 설정 하에서 반복하여 파이프라인을 실행
  - 데이터 설정에 따라 일부 프로세스 변경이 필요한 경우, 해당 프로세스만 수정 후 전체 파이프라인을 일괄 수행

# Pipeline - robustness check

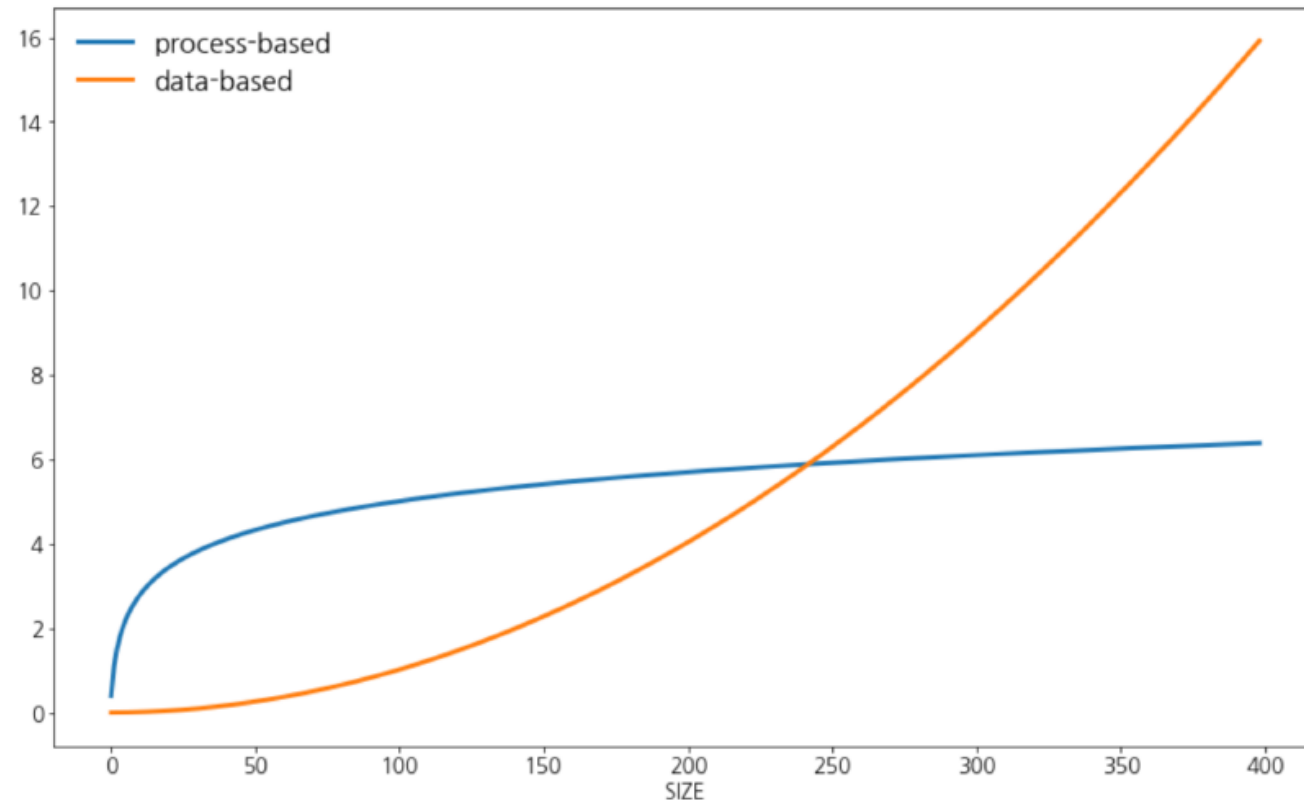
---

- 단 하나의 변수 및 표본 구성, 계량모형에서 얻은 분석 결과는 동 표본에서만 유효할 가능성이 매우 높음
  - 변수 예측을 최우선 목적으로 하는 머신러닝에서는 표본 내 예측력은 높지만 표본 외 예측력이 낮은 과적합 (overfitting) 방지가 최우선 과제
  - 일반적인 계량분석에서는 분석결과가 이론에 부합하고, 추정계수가 유의하면 강건성을 문제삼는 경우가 적음
- (어쩌다 얻어 걸린) 분석결과를 과도하게 신뢰하여 오랜기간 정성들여 보고서를 작성하고 나면 이후 되돌이킬 수 없으므로, 보고서 작성 전에 다양한 변수 조합과 변환, 표본기간에 대해 유사한 분석 결과가 얻어지는지 확인



# Pipeline vs. Excel

- 엑셀 기반 데이터 분석에 소요되는 시간은 데이터 크기에 따라 급격하게 증가하지만, 파이프라인 기반 데이터 분석의 경우 데이터 크기의 영향을 거의 받지 않음



# 데이터

---

- Data formats - Excel, 이미지 등
- Aggregate and granular/micro data - 월별 집계 데이터, 개별 거래기록 등
- Data sources - 설문조사, 공공데이터
- Preprocess, cleansing, process - 결측치, 이상치, 데이터 변환(정규화 등)
- Data analysis methods - EDA, 예측, 분류 등
- Programming languages/Data tools - python, R 등

# 데이터 포맷

---

- Xlsx, csv, dta, mat, txt, json, xml, pickle, ...  
download from webpages, web crawling, api, libraries, ...
- Structured: time series, cross-sectional, panel
  - Table 형태로 주어짐
- Unstructured: text, audio, video, images
  - Table로 변환하여 분석

# Aggregate vs. granular/micro data

- To infer the relation between income and consumption from aggregate and granular data

$t$	$C$	$I$
2010	1.32	0.81
2011	2.51	0.93
2012	1.13	1.02
$\vdots$	$\vdots$	$\vdots$

$\Downarrow$

$$E[C|I] = \beta_0 + \beta_1 I$$

V.

$t$	$i$	$C$	$I$
2010	1	1.45	0.85
2011	1	2.82	0.94
2012	1	1.45	1.03
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2010	2	0.87	0.79
2011	2	2.31	0.90
2012	2	1.03	1.01
$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\Rightarrow$

$E[C I^*]$	$I^*$
1.45	(0.7, 0.8]
1.51	(0.8, 0.9]
1.55	(0.9, 1.0]
$\vdots$	$\vdots$

# Sources

---

- ECOS, MDSS, BIDAS, FAIRS, FISS, 가계부채DB, 외환전산망 등
- KOSIS, KIS-value, NICE, KCB, KRX, 신용정보원 등
- IMF, BIS, FRED, OECD, UN, World Bank, WTO, 각국 중앙은행 등
- 노동연구원, 재정연구원, 무역협회, 고용노동부, 한국생산성본부, 서울시
- Bloomberg, Refinitiv, Infomax
- Google, 온라인 쇼핑몰, 인터넷 뉴스, 포털 사이트, 카드사 등

# Import, preprocess, cleansing

---

- 데이터 입수
- 날짜 포맷, 데이터 타입, 개체명 등 표준화
- 통계량, 결측치, 이상치 제거
- 데이터 선택, 결합
- 변수 생성, 통계량 계산, 시계열 빈도 변환
- 데이터 구조 설정
- 시각화

# Analysis/visualization

---

- 기초 통계 분석(Descriptive statistics)
- Regression and Econometric Models - regression, VAR, local projection, state-space model
- Advanced Economic Models - DSGE, ABM, OLG
- Machine learning & NLP
- Data Visualization Techniques - 기본(Line, bar, pie charts), 고급(Network, heatmap, word cloud) 등

# Analysis - regression

---

- Q: x와 y의 관계는?
- 여타 요인(Z)을 통제한 두 변수 간 상관계수( $\beta$ )

$$y = \alpha + \beta x + \gamma z + \epsilon$$

- 인과관계(causality)를 이야기하기 위해서는
  - x, y, z(x, y와 관련있는 주요 변수)를 포함한 설득력 있는 모형을 찾고,
  - 동 모형에 근거한 계량모형을 이용하여 회귀계수를 추정
- 한은소식 ‘상관과 회귀’ (한은소식(2022년 6월호))



# Analysis - machine learning

---

- 계량경제 모형은 변수의 확률분포에 대한 가정 하에 변수 간 관계 추정에 초점
  - 변수 간 관계는 통산 선형 회귀식으로 표현
  - 데이터를 모두 사용하여 회귀계수를 추정
  - (모형검증) 추정결과가 이론에 부합하는지 여부,  $R^2$ , AIC, BIC 등으로 유효성 판단
- 머신러닝 알고리즘은 확률분포에 대한 가정없이 변수 예측에 초점
  - 특정 모형구조(하이퍼파라미터) 하에서 변수 간 관계를 포착하는 패턴(파라미터)을 알고리즘적으로 탐색
  - 일부 데이터에 대해 패턴을 학습한 다음 나머지 데이터를 이용하여 표본 외 예측력 평가
  - (모형검증) 표본외 예측력을 기준으로 모형의 유효성을 판단

# Analysis - machine learning

---

- 변화에 대처하는 슬기로운 한국은행(한은소식 2021년 8월호)
  - 2016년 알파고, ‘머신러닝은 블랙박스라서 안돼’
  - 표본 외 예측력만 좋다면 interpretability vs. explainability

# Programming languages/data tools

tool	data process	analysis	visualization	community	accessibility
Stata	◐	●	◐	◐	◐
Matlab	◐	●	●	◐	◐
Gauss	○	◐	○	○	◐
E-views	○	○	○	○	◐
SAS	◐	◐	○	○	○
C/Fortran	○	◐	○	◐	○
R	●	◐	●	◐	◐
Python	●	●	●	●	●
Julia	◐	●	◐	◐	●

Note: ● is better than ○.

**빅데이터**

# 빅데이터

---

- 5 Vs(velocity, volume, value, variety, and veracity)

- 1) **Velocity (속도)**

- 데이터 생성 및 처리 속도. 실시간 데이터 스트림 분석 필요.
    - 예시: 카드 결제 트랜잭션, IoT 센서 데이터, 금융 거래, 국고채 거래

- 2) **Volume (규모)**

- 방대한 양의 데이터.
    - 예시: 소셜 미디어, 전 세계 일일 검색 데이터

- 3) **Value (가치)**

- 데이터 활용으로 얻는 비즈니스 및 사회적 가치.
    - 예시: 금융 모델 최적화, 소비자 행동 분석.

- 4) **Variety (다양성)**

- 데이터 형식: 정형(Structured), 반정형(Semi-Structured), 비정형(Unstructured).
    - 예시: 텍스트, 이미지, 영상 데이터.

- 5) **Veracity (신뢰성)**

- 데이터의 정확성과 품질.
    - 예시: 가짜 뉴스 필터링, 데이터 클렌징 필요성.

# 빅데이터

---

- 데이터기반 정책 사례

**BIS working paper ‘Big data and machine learning in central banking’**

(<https://www.bis.org/publ/work930.pdf>)

**과학기술부 통합 데이터 지도(금융, 유통, 지역경제 등 16개 빅데이터 플랫폼)**

(<https://www.bigdata-map.kr/>)

**BIDAS - BigData Hub**

# 주요 빅데이터 및 활용 사례

---

- 대규모 경제금융 시계열(GDP nowcasting, 조기경보모형)
- 거래 데이터(금융시장 이상탐지, 주택시장 실거래가격지수 등)
- 가계부채DB(신용위험 평가, 업권·차주·빈티지별 연체율 등)
- 신용카드, 전력사용량, 교통 등(경기 모니터링)
- 텍스트, 비디오/이미지, 오디오 등 비정형 데이터(전체 웹 데이터의 80~90%)
  - 인공지능 언어모형(인플레이션 어조지수 등)

# 파이썬과 주피터노트북



# 디지털신기술과 데이터 분석

---

- 다양한 형태의 빅데이터는 보다 효율적인 데이터 수집, 전처리, 가공 및 분석, 시각화 역량을 요구
  - 새로운 데이터 환경 및 분석 수요에 효과적으로 대응하기 위해서는
    1. 탐색적 데이터 분석과 협업이 용이한 환경에서
    2. 여러 데이터 도구를 효과적으로 결합하여 활용할 수 있는
    3. 사용자 인터페이스와 데이터 분석 언어를 선택
- > 주피터노트북과 파이썬





















# 파이썬

---

- 접근성(open source)
- 다양한 라이브러리로 높은 생산성(확장성, 유연성, 호환성으로 서버, 네트워크, 시스템 등 다양한 영역에서 사용)
- 방대한 사용자 커뮤니티(github, stackoverflow, ...)
- 인터프리터 (Interpreter) 언어
- 간결성(pythonic way of coding, philosophy)
- 객체지향
- 편리한 인터페이스(Jupyter notebook/lab)
- 딥러닝 프레임워크인 텐서플로(Tensorflow), 케라스(Keras), 파이토치(PyTorch) 등에서 파이썬 우선 지원

# 파이썬

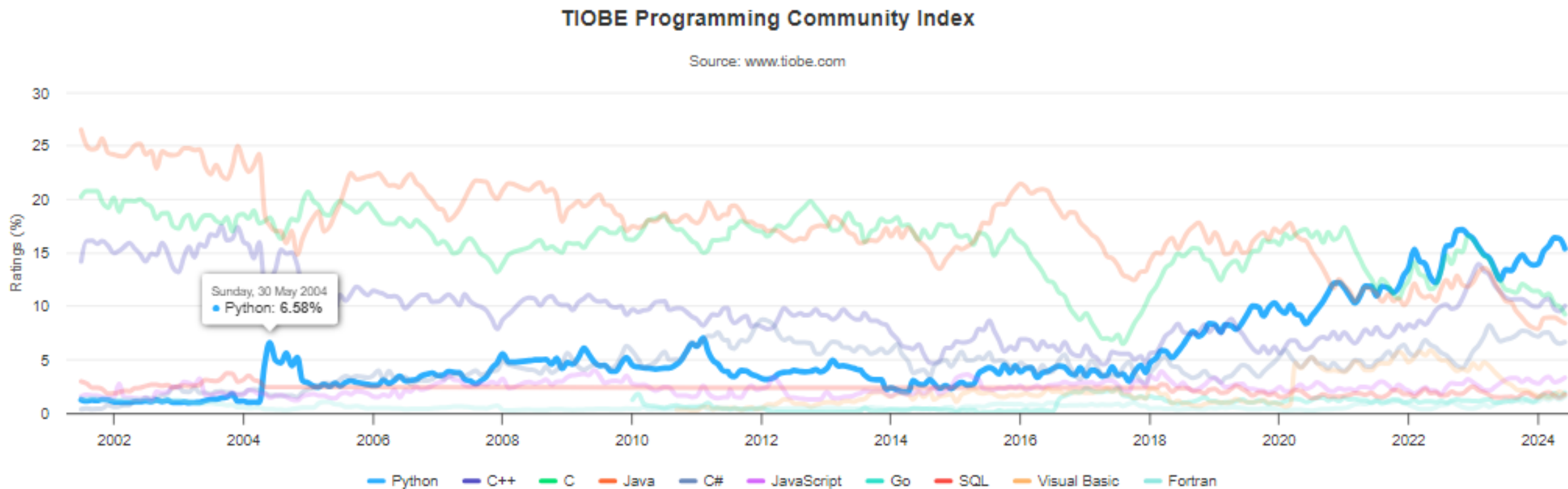
- TIOBE Index for June 2024 (# of skilled engineers, coursed available, and etc...)

Jun 2024	Jun 2023	Change	Programming Language		Ratings	Change
1	1		 Python		15.39%	+2.93%
2	3	▲	 C++		10.03%	-1.33%
3	2	▼	 C		9.23%	-3.14%
4	4		 Java		8.40%	-2.88%
5	5		 C#		6.65%	-0.06%
6	7	▲	 JavaScript		3.32%	+0.51%
7	14	▲	 Go		1.93%	+0.93%
8	9	▲	 SQL		1.75%	+0.28%
9	6	▼	 Visual Basic		1.66%	-1.67%
10	15	▲	 Fortran		1.53%	+0.53%
11	11		 Delphi/Object Pascal		1.52%	+0.27%
12	19	▲	 Swift		1.27%	+0.33%
13	10	▼	 Assembly language		1.26%	-0.03%
14	12	▼	 MATLAB		1.26%	+0.14%
15	8	▼	 PHP		1.22%	-0.52%
16	13	▼	 Scratch		1.17%	+0.15%
17	20	▲	 Rust		1.17%	+0.26%
18	18		 Ruby		1.11%	+0.17%
19	29	▲	 Kotlin		1.01%	+0.50%
20	22	▲	 COBOL		0.96%	+0.22%

Position	Programming Language	Ratings
21	R	0.96%
22	SAS	0.92%
23	Dart	0.89%
24	Prolog	0.80%
25	Ada	0.80%
26	D	0.75%
27	Perl	0.69%
28	Classic Visual Basic	0.62%
29	Haskell	0.59%
30	(Visual) FoxPro	0.57%
31	Scala	0.54%
32	Lua	0.51%
33	Julia	0.48%
34	GAMS	0.45%
35	ML	0.44%
36	Lisp	0.44%
37	Objective-C	0.44%
38	Transact-SQL	0.40%
39	VBScript	0.36%
40	ABAP	0.33%
41	PowerShell	0.33%
42	Scheme	0.32%
43	Bash	0.32%
44	Logo	0.29%
45	LabVIEW	0.27%
46	F#	0.27%
47	Solidity	0.27%
48	Awk	0.27%
49	PL/SQL	0.26%
50	TypeScript	0.26%

# 파이썬

- TIOBE Index for June 2024 (# of skilled engineers, coursed available, and etc...)



# 파이썬으로 할 수 있는 것

---

- Matlab 대체
- Stata 보완
- 머신러닝
- 텍스트 분석
- 파이프라인 구축(MLOps)
- Agent-based model 등

# 파이썬 - Matlab, Stata

---

- Matlab으로 작성된 코드는 대부분 파이썬으로 변환하여 보다 효율적으로 실행할 수 있음
- 파이썬으로 작성된 코드가 매틀랩에 비해 느리게 실행되는 경우 Cython을 이용하여 매틀랩보다 빠르게 실행할 수도 있음(Kalman filter)
- 계량경제모형을 이용할 경우 Stata와 파이썬을 보완적으로 사용

# 파이썬 - ML and MLOps

---

- 대부분 머신러닝 라이브러리가 파이썬을 지원
  - Scikit-learn
  - pytorch
  - tensorflow
  - huggingface
- 데이터 입수, 전처리, ML, 시각화 전 과정을 효율적으로 관리

# 파이썬 - contextual data analysis

---

- Table 형태 데이터의 각 행과 열에 의미있는 이름(label)을 부여하여 전 데이터 분석 프로세스(파이프라인)를 효율적으로 관리
  - 데이터 선택, 결합, 연산 프로세스를 오류없이 빠르게 수행
  - 가독성이 높아지므로 분석 결과 업데이트, 공유가 용이
- 행과 열 label은 날짜, 문자열, 숫자 등으로 지정
  - 파이썬의 강력한 날짜, 문자열 처리 기능을 이용하여 손쉽게 라벨을 생성
  - 데이터분석 라이브러리 Pandas의 계층 라벨은 보다 구조화된 분석을 지원



# 파이썬 - contextual data analysis

---

		Country A			Country B		
		Sector 1	Sector 2	Final Consumption	Sector 1	Sector 2	Final consumption
Country A	Sector 1	0	20	5	0	30	10
	Sector 2	0	0	10	0	0	10
Country B	Sector 1	0	0	0	0	20	20
	Sector 2	0	0	20	0	0	30

# 주피터노트북

---

- 직관적이고 편리한 웹브라우저 기반 인터페이스
  - 일련의 입력과 출력, 텍스트와 코드 셀로 구성
  - 데이터 입수부터 가공, 분석, 시각화, 보고서/주석 전 과정 관리
- 다양한 분석환경에서 이용 가능
  - BIDAS, VS code, Google Colab
- 효율적인 파이프라인 구축 및 공유
  - 재현가능연구(reproducible research)

# 주피터노트북 - 화면 예시

The screenshot displays a Jupyter Notebook interface in a web browser. The browser's address bar shows 'localhost'. The notebook's title bar indicates 'python2020/4계투입산출표'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. On the right, there are buttons for 'Trusted' and 'Python 3'.

The main content area shows a code cell with the following Python code:

```
In [95]: import pandas as pd
import numpy as np
from IPython.display import IFrame, Image

idx = pd.IndexSlice
```

Below the code cell, the text '레온티에프 역행렬 함수' is displayed. Underneath, the text '투입산출표' is shown. This is followed by the text '중간투입행렬 Z, 최종수요행렬 F, 부가가치벡터 V, 총산출벡터 X'.

The mathematical formula for the Leontief inverse is presented:

$$\begin{bmatrix} Z & F \\ V' & \\ X' & \end{bmatrix}, \text{ where } Z = \begin{bmatrix} Z_{c-m,c-m} & Z_{c-m,c-s} & Z_{c-m,k-m} & Z_{c-m,k-s} \\ Z_{c-s,c-m} & Z_{c-s,c-s} & Z_{c-s,k-m} & Z_{c-s,k-s} \\ Z_{k-m,c-m} & Z_{k-m,c-s} & Z_{k-m,k-m} & Z_{k-m,k-s} \\ Z_{k-s,c-m} & Z_{k-s,c-s} & Z_{k-s,k-m} & Z_{k-s,k-s} \end{bmatrix} \text{ and } F = \begin{bmatrix} F_{c-m,c-c} & F_{c-m,c-i} & F_{c-m,k-c} & F_{c-m,k-i} \\ F_{c-s,c-c} & F_{c-s,c-i} & F_{c-s,k-c} & F_{c-s,k-i} \\ F_{k-m,c-c} & F_{k-m,c-i} & F_{k-m,k-c} & F_{k-m,k-i} \\ F_{k-s,c-c} & F_{k-s,c-i} & F_{k-s,k-c} & F_{k-s,k-i} \end{bmatrix}$$

Below the formula, the text '레온티에프 역행렬:  $(I - A)^{-1}$ ' is shown. This is followed by the text 'X = AX + f with  $A_{i,j} = Z_{i,j}/X_j$  and  $f_i = \sum_j F_{i,j}$ '.

The text 'X = (I - A)<sup>-1</sup>f' is displayed.

Below the text, a code cell shows the following Python code:

```
In [96]: def Leon(A):
return pd.DataFrame(np.linalg.inv(np.identity(A.shape[0]) - A),
                    index=A.index, columns=A.columns)
```

Below the code cell, the text '주피터노트북에서 수식 편집' is displayed.

Below the text, a code cell shows the following Python code:

```
In [97]: url = ('http://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Typesetting%20Equations.html')
IFrame(url, width=900, height=450)
```

Below the code cell, the text 'Out[97]:' is displayed. This is followed by a list of links: 'Motivating Examples', 'The Lorenz Equations', 'Docs', 'Notebook Examples', 'Motivating Examples', and 'Edit on GitHub'.

# 파이썬 라이브러리

---

- 모듈
- 패키지
- 라이브러리
- 프레임워크

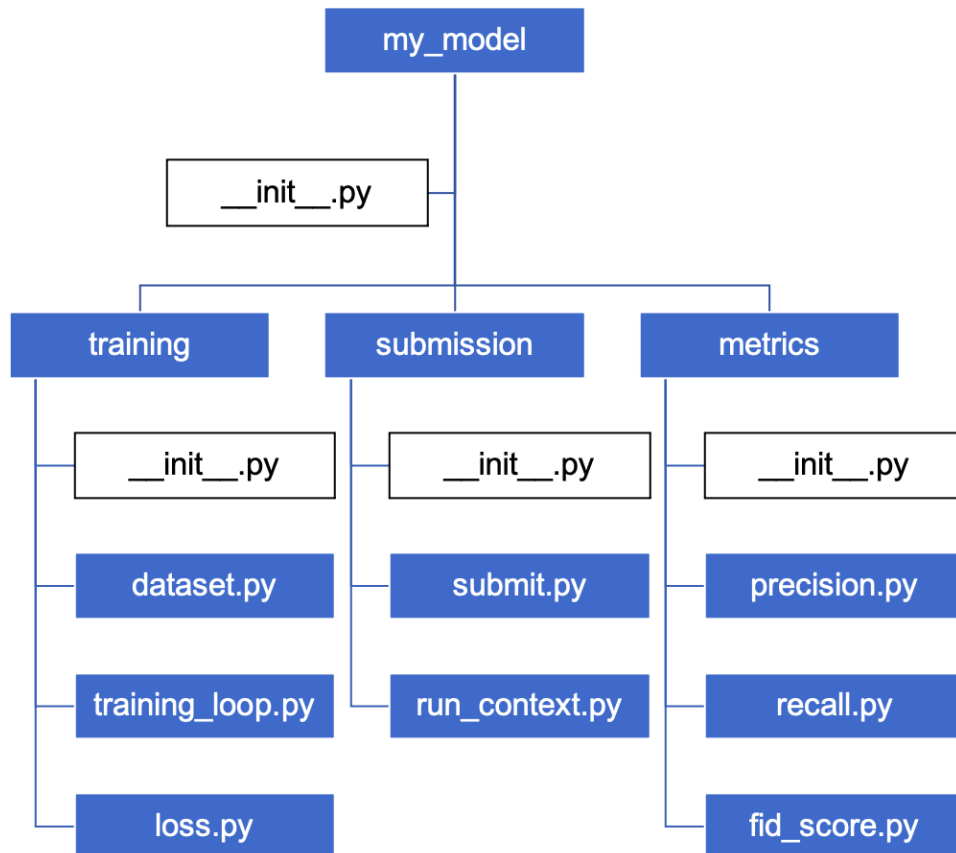
# 파이썬 모듈

---

- 모듈은 기본적으로 관련된 코드가 .py 확장자로 저장된 파일
  - `random` module to generate pseudo-random number generators for various distributions.
  - `html` module to parse HTML pages.
  - `datetime` module to manipulate date and time data.
  - `re` module to detect and parse regular expressions in Python.

# 파이썬 패키지

- Python packages are basically a directory of a collection of modules



- NumPy** is the fundamental Python package for scientific computing.
- pandas** is a Python package for fast and efficient processing of tabular data, time series, matrix data, etc.
- pytest** provides a variety of modules to test new code, including small unit tests or complex functional tests.

# 파이썬 라이브러리

---

- 라이브러리는 재사용 가능한 코드 조각을 지칭하는 포괄적인 용어입니다.
- 일반적으로 Python 라이브러리는 관련된 모듈과 패키지의 모음으로 구성됩니다.

- 'Python 패키지'와 혼용되어 사용되는 경우가 多

By the way, the **NumPy** and **pandas** packages that were mentioned before are also often referred to as libraries. That is because these are complex packages that have wide applications (i.e. scientific computing and data manipulation, respectively). They also include multiple subpackages and so basically satisfy the definition of a Python library. Learn about other important libraries for data science in [THIS ARTICLE](#).

- **Matplotlib** library is a standard library for generating data visualizations in Python. It supports building basic two-dimensional graphs as well as more complex animated and interactive visualizations.
- **PyTorch** is an open-source deep-learning library built by Facebook's AI Research lab to implement advanced neural networks and cutting-edge research ideas in industry and academia.
- **pygame** provides developers with tons of convenient features and tools to make game development a more intuitive task.
- **Beautiful Soup** is a very popular Python library for getting data from the web. The modules and packages inside this library help extract useful information from HTML and XML files.
- **Requests** is a part of a large collection of libraries designed to make Python HTTP requests simpler. The library offers an intuitive JSON method that helps you avoid manually adding query strings to your URLs.
- **missingno** is very handy for handling missing data points. It provides informative visualizations about the missing values in a dataframe, helping data scientists to spot areas with missing data. It is just one of the many great [Python libraries for data cleaning](#).

# 파이썬 프레임워크

---

- 라이브러리와 유사하게, Python 프레임워크는 개발 과정을 빠르게 진행할 수 있도록 돕는 모듈과 패키지  
의 모음입니다.
- Python 프레임워크는 개발을 더 빠르고 쉽게 할 수 있도록 도와주는 도구로 프레임워크는 보통 더 복잡하  
고 규모가 큼니다
  - **Django** is a Python framework for building web applications with less coding. With all the necessary features included by default, developers can focus on their applications rather than dealing with routine processes.
  - **Flask** is a web development framework that is known for its lightweight and modular design. It has many out-of-the-box features and is easily adaptable to specific requirements.
  - **Bottle** is another lightweight framework for web development that was originally meant for building APIs. Its unique features are that it has no dependencies other than the **Python Standard Library** and it implements everything in a single source file.



# 파이썬 라이브러리

- 데이터 분석 & 시각화: numpy, pandas, matplotlib, seaborn, scipy
- 머신러닝: sklearn, tensorflow, pytorch, keras, nltk
- 웹개발: django, flask, dash, request

특징	NumPy	Pandas
주요구조	N차원 배열 (ndarray)	데이터프레임 (DataFrame) 및 시리즈 (Series)
데이터타입	동일 타입 (숫자 데이터에 최적화)	다양한 타입 (숫자, 문자열, 날짜 등)
주요기능	수학/과학 계산, 배열 연산	데이터 조작, 필터링, 그룹화, 결측치 처리
입출력지원	제한적 (파일 읽기/쓰기는 Pandas 사용 권장)	다양한 파일 포맷 지원 (CSV, Excel 등)
사용대상	고속 계산이 필요한 수치 데이터 처리	표 형식 데이터를 다루는 데이터 분석 작업

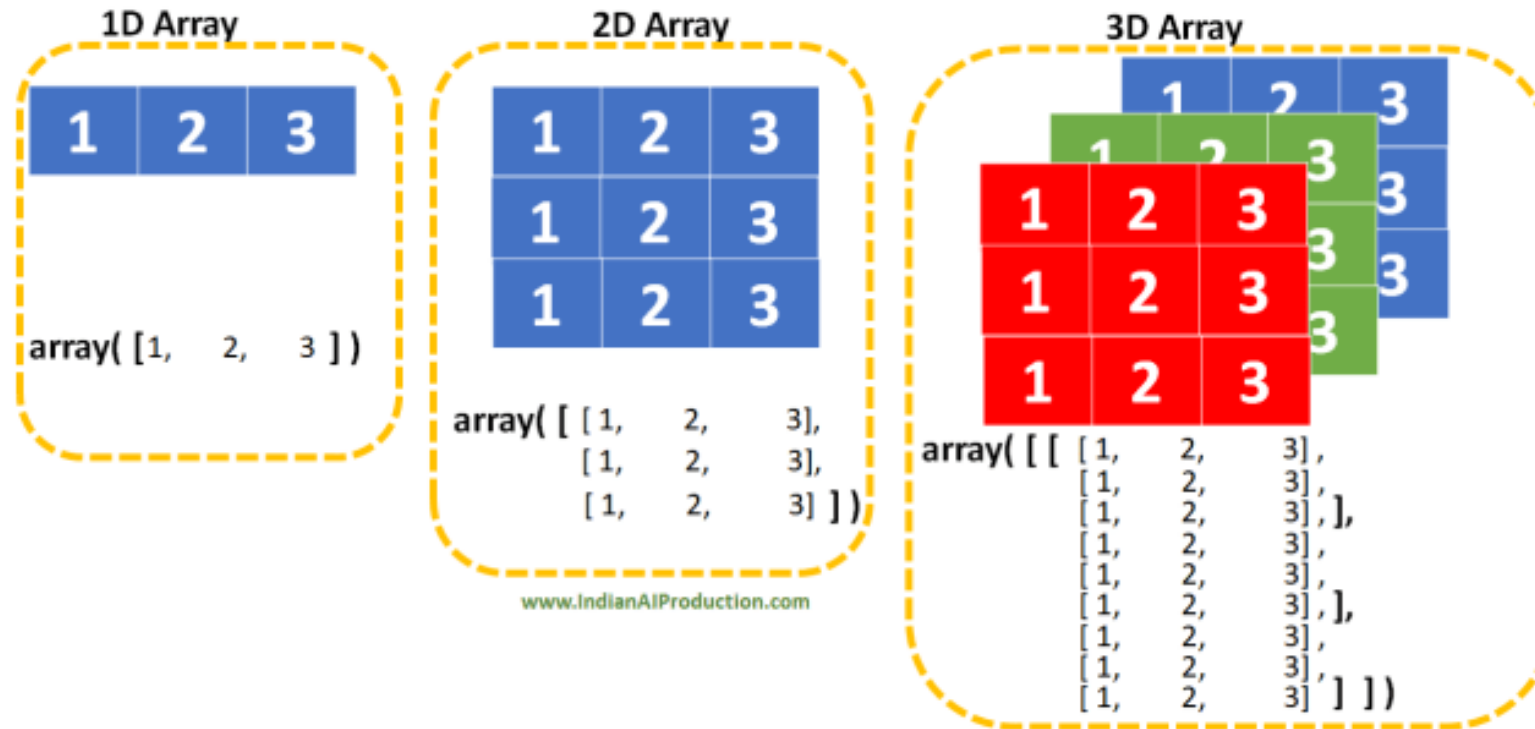
# Numpy(Numerical Python)

---



- 선형대수 기반의 프로그램을 쉽게 만들 수 있도록 지원하는 패키지
- Numpy의 핵심 함수들이 C언어로 구현되어 있어서 빠름
- 벡터화, BLAS(Basic Linear Algebra Subprograms)와 LAPACK(Linear Algebra Package) 등 최적화된 수학 라이브러리 사용
- 파이썬에서 반복문을 사용하지 않고 대량 데이터 연산을 가능하게 함

# Numpy(Numerical Python)



# Numpy(Numerical Python)

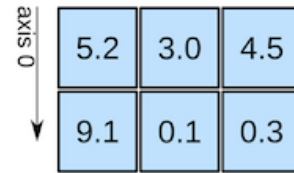


1D array



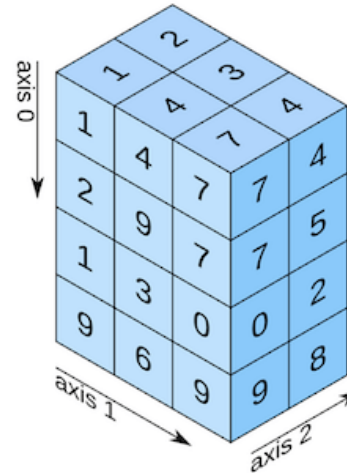
shape: (4,)

2D array



shape: (2, 3)

3D array

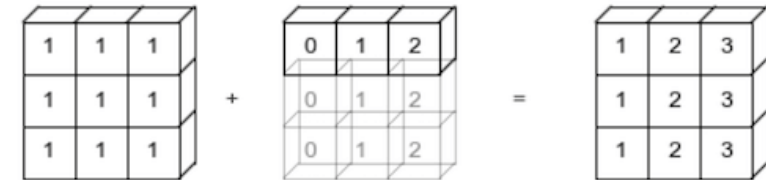


shape: (4, 3, 2)

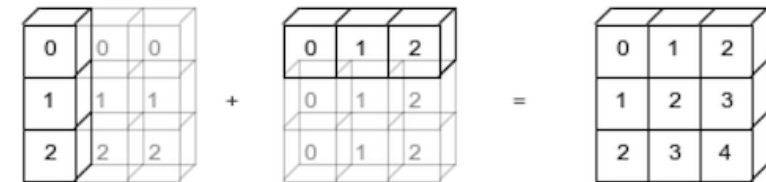
`np.arange(3)+5`



`np.ones((3,3))+np.arange(3)`



`np.arange(3).reshape((3,1))+np.arange(3)`



## Jupyter format

	YEARMODA	TEMP	MAX	MIN
0	20160601	65.5	73.6	54.7
1	20160602	65.8	80.8	55.0
2	20160603	68.4	77.9	55.6
3	20160604	57.5	70.9	47.3
4	20160605	51.4	58.3	43.2
5	20160606	52.2	59.7	42.8
6	20160607	56.9	65.1	45.9
7	20160608	54.2	60.4	47.5
8	20160609	49.4	54.1	45.7
9	20160610	49.5	55.9	43.0

## Pandas DataFrame

`pandas.core.frame.DataFrame`

## Standard Python format

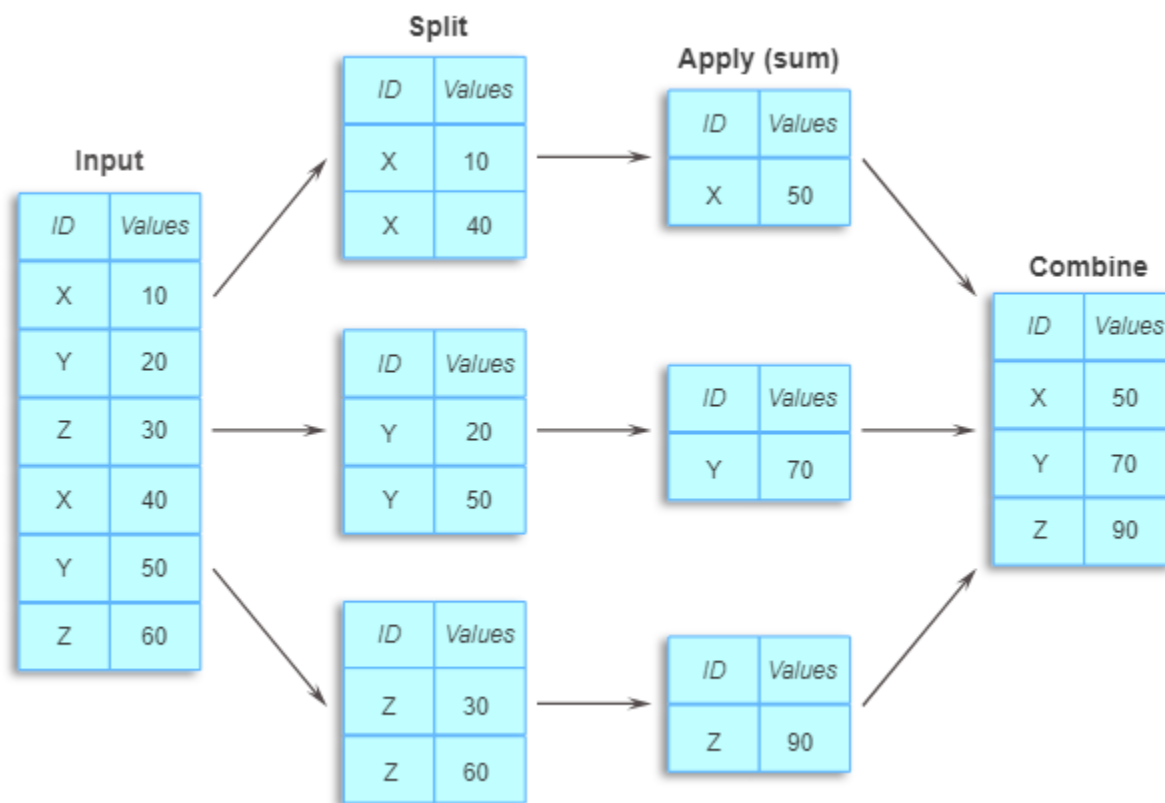
	YEARMODA	TEMP	MAX	MIN
0	20160601	65.5	73.6	54.7
1	20160602	65.8	80.8	55.0
2	20160603	68.4	77.9	55.6
3	20160604	57.5	70.9	47.3
4	20160605	51.4	58.3	43.2
5	20160606	52.2	59.7	42.8
6	20160607	56.9	65.1	45.9
7	20160608	54.2	60.4	47.5
8	20160609	49.4	54.1	45.7
9	20160610	49.5	55.9	43.0

## Standard Python format

```
0    65.5
1    65.8
2    68.4
3    57.5
4    51.4
5    52.2
6    56.9
7    54.2
8    49.4
9    49.5
Name: TEMP, dtype: float64
```

## Pandas Series

`pandas.core.series.Series`



# 실습

---

- <https://github.com/pjhbrain/python101>

**머신러닝**



- AI: 인간의 학습, 추론 능력을 갖춘 컴퓨터 시스템 또는 관련된 과학기술
  - (strong AI) 사람처럼 행동하고 다양한 업무를 수행
  - (weak AI) 좁은 범위, 또는 단일한 업무를 처리하는 인공지능
- ML: 인공지능을 구현하기 위한 기술
  - “T(task)라는 작업을 수행하는 어떤 컴퓨터 프로그램의 성능 P(performance measure)가 E(experience)를 통해 향상된다면 이 프로그램은 E를 통해 학습한다” Mitchell(1997)
  - “명시적인 프로그래밍이나 지시 없이 데이터의 패턴을 인식하는 기법”

# Learning

---

- 지도학습 supervised learning
  - 입력변수와 라벨로 구성된 데이터를 통해 학습
  - 예측(분류, 회귀)
  - 인공지능 언어모형 fine-tuning
- 비지도학습 unsupervised learning
  - 인공지능 언어모형 pre-training
- 강화학습 reinforcement learning
  - 알파고

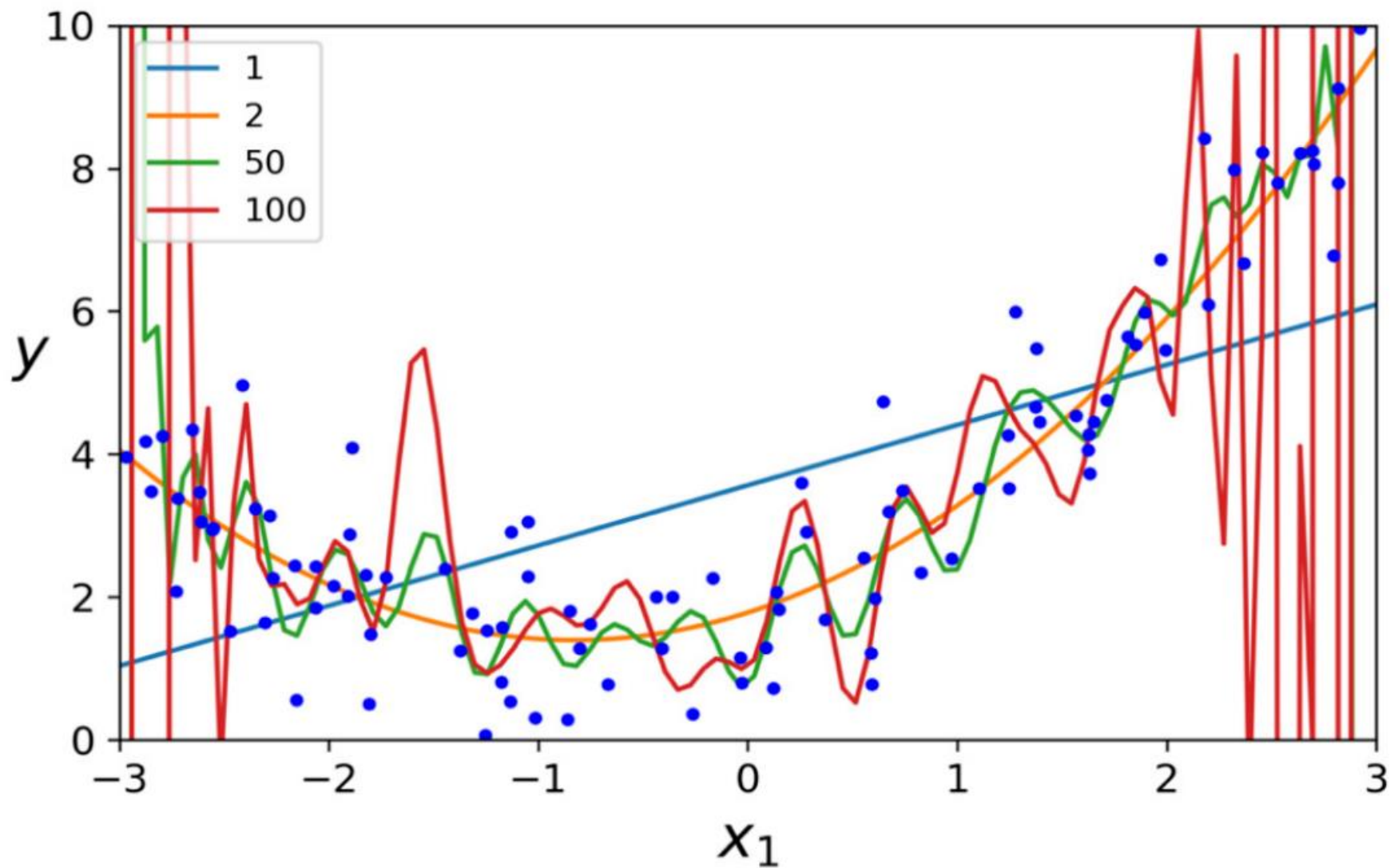
# Hyperparameter

---

- 하이퍼파라미터는 머신러닝 알고리즘이 데이터의 패턴을 학습하는 방식, 파라미터는 머신러닝 알고리즘이 패턴을 인식한 결과를 의미
  - AR 모형의 하이퍼파라미터는 모형의 최대 시차, 파라미터는 AR 모형의 시차별 계수
- 하이퍼파라미터 최적화를 통해 머신러닝 알고리즘의 표본외 예측력을 극대화
  1. 전체 데이터를 학습 데이터와 테스트 데이터로 구분
  2. 학습 데이터에 대해 하이퍼파라미터를 바꿔가며 머신러닝 알고리즘을 학습
  3. 테스트 데이터를 이용하여 각(하이퍼파라미터 별) 알고리즘의 예측력을 계산하고 예측력이 가장 높은 하이퍼파라미터 선택

# Overfitting

박기영 · 고정원(2019)



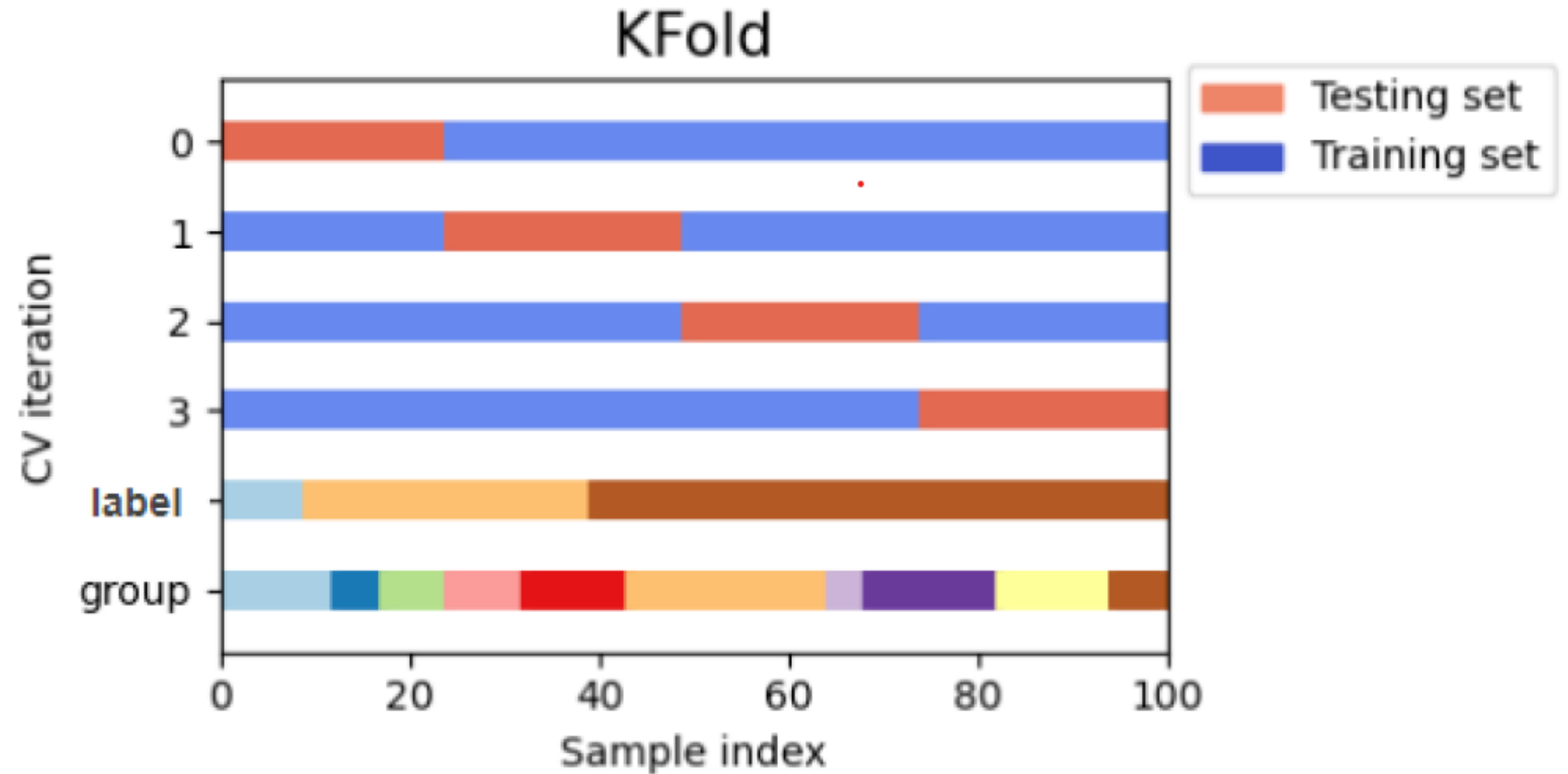
# Model validation techniques

---

- 경제이론,  $R^2$ , *Accuracy* 등
  - Holdout(train/test split) validation
  - Leave-one-out cross-validation
  - K-fold cross-validation
  - Walk-forward(time series split) validation
- > 데이터 속성(그룹)을 고려하여 과적합을 최소화하는 기법 적용

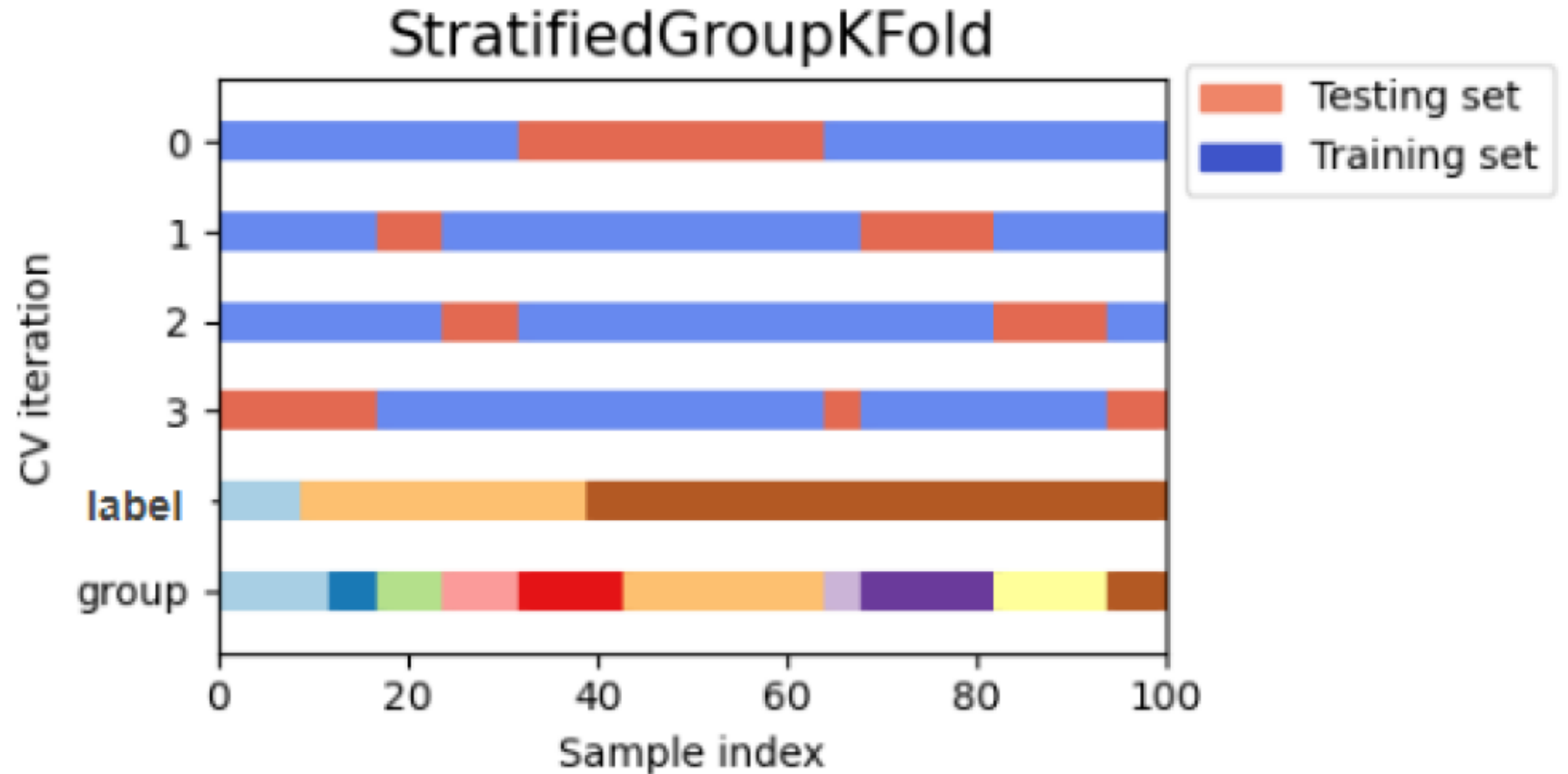
# K-fold cross-validation

- 100개 관측치
- 3개 label
- 10개 group
- 관측 순서대로  
fold(training/test  
datasets) 구성



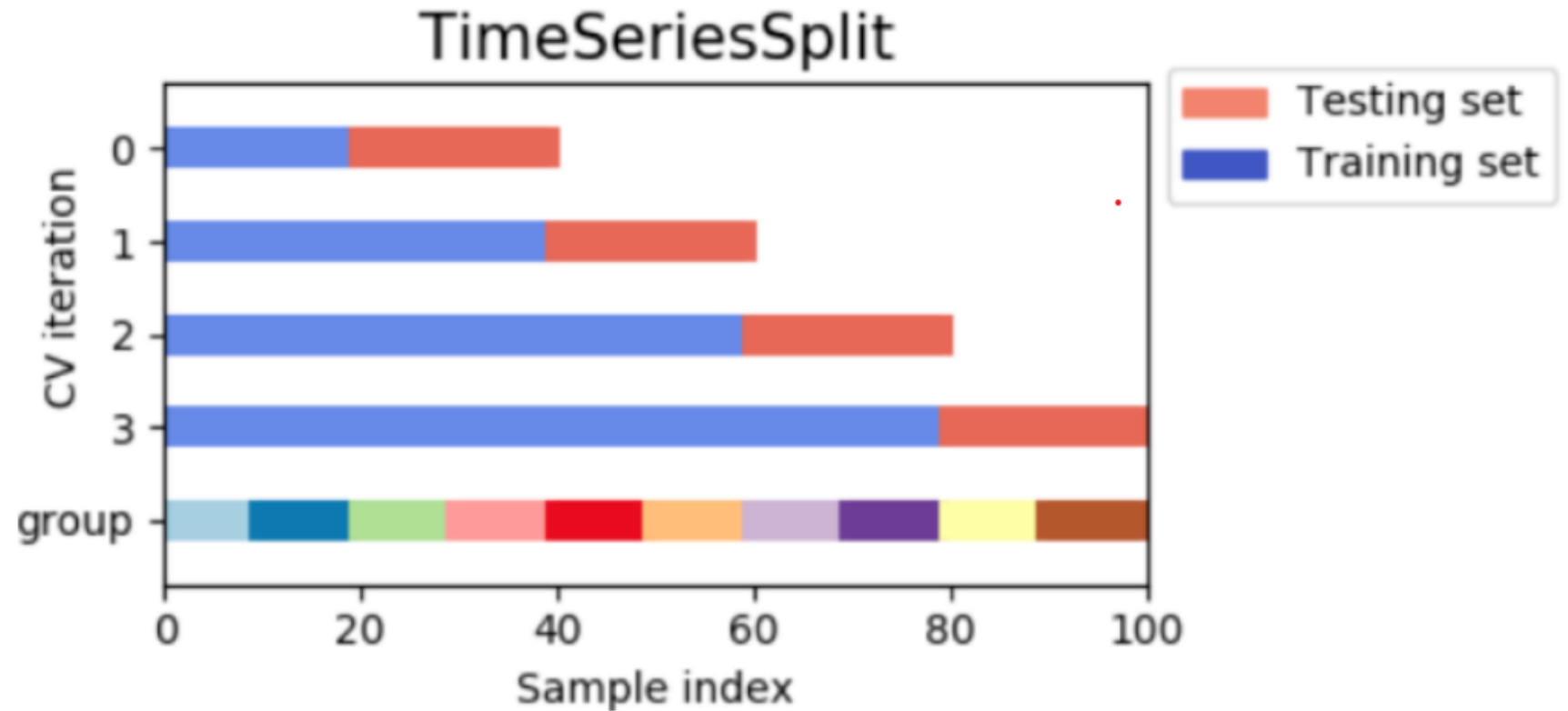
# Stratified Group K-fold cross-validation

- 100개 관측치
- 3개 label
- 10개 group
- Group 단위로 각 label을 골고루
- 조기경보모형



# Walk-forward validation

- 100개 관측치
- 10개 group
- 시간순서대로  
training set 구성
- GDP nowcasting





# ML Algorithms

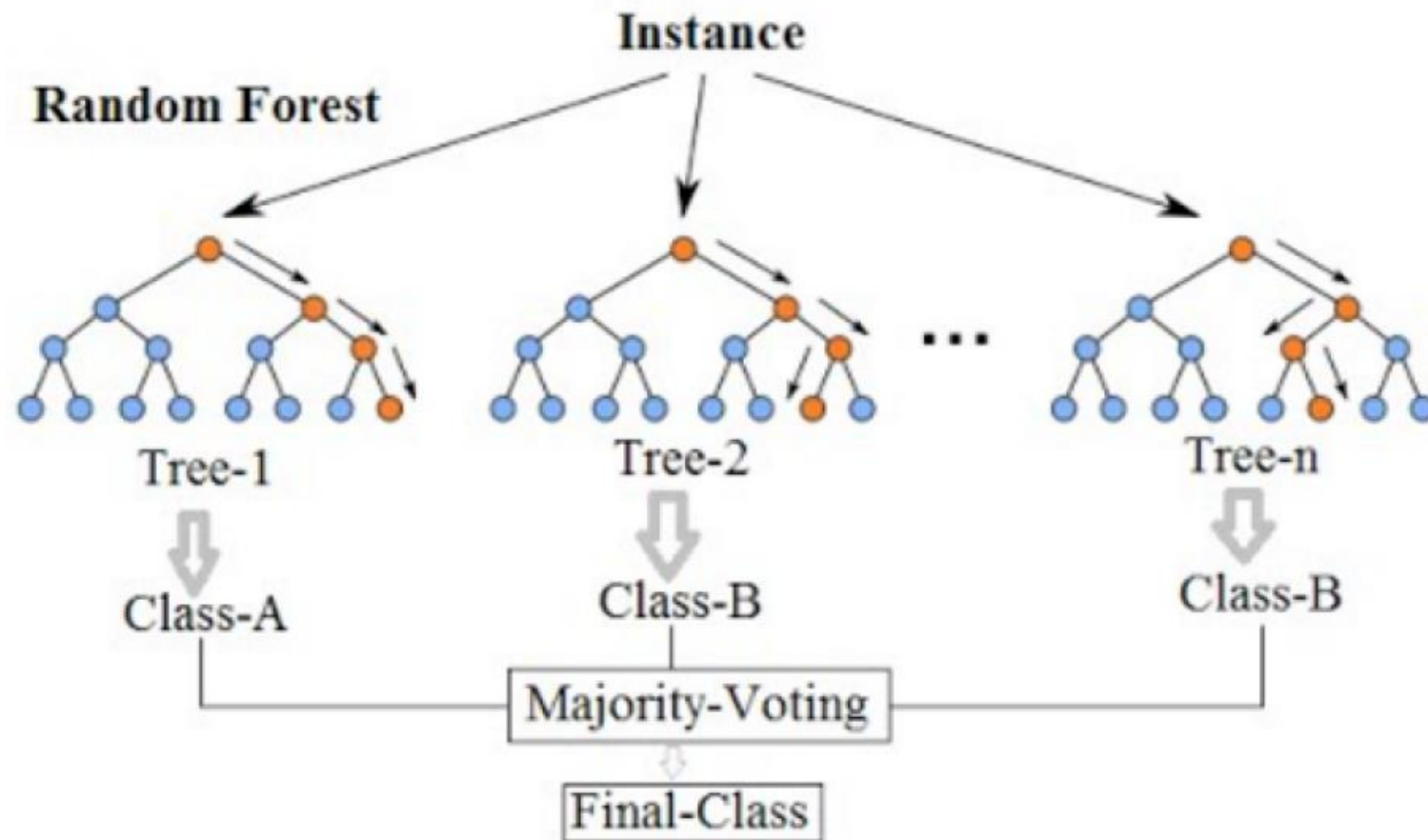
---

- 기본적인 Regression
  - Lasso, Ridge, Elastic-net
- Classification 및 Regression - SVM, Decision Tree
- Ensemble
  - 51% 정확도의 분류기 1,000개로 75%의 정확도 달성
  - Random Forest, Extremely Randomized Trees
- Deep learning
  - NN, RNN/LSTM, CNN

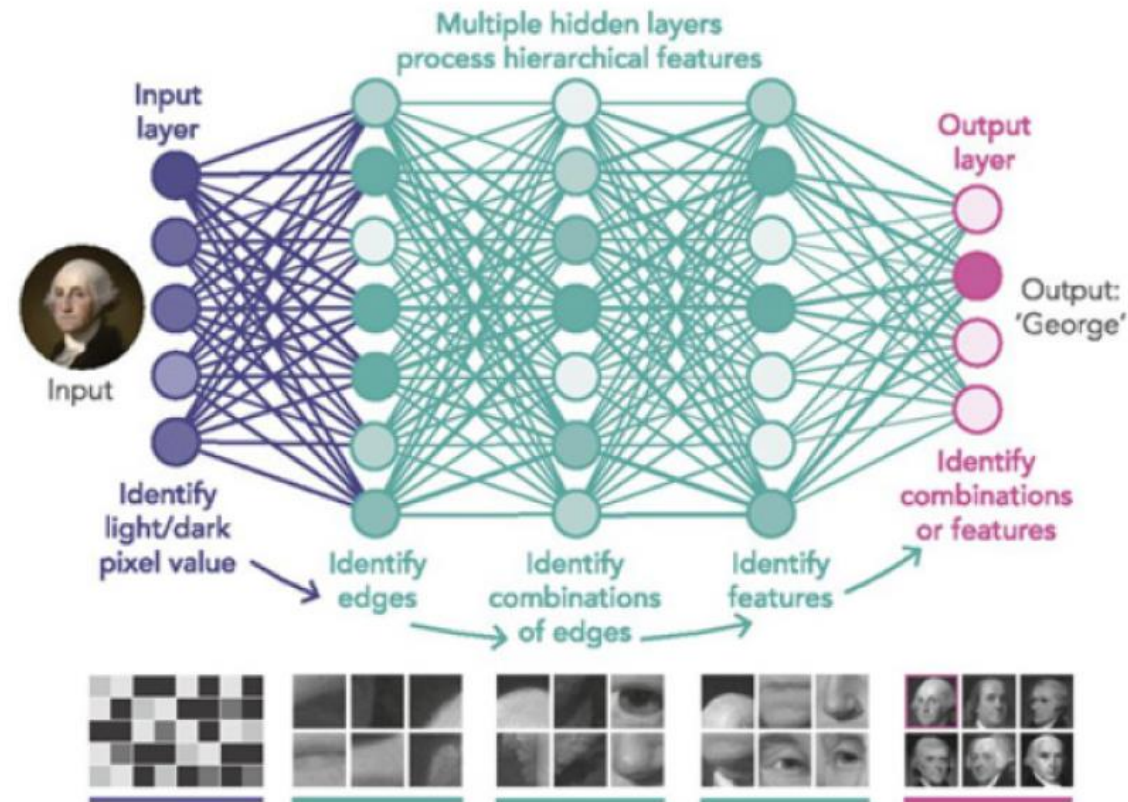
# Random Forest

---

## Random Forest Simplified



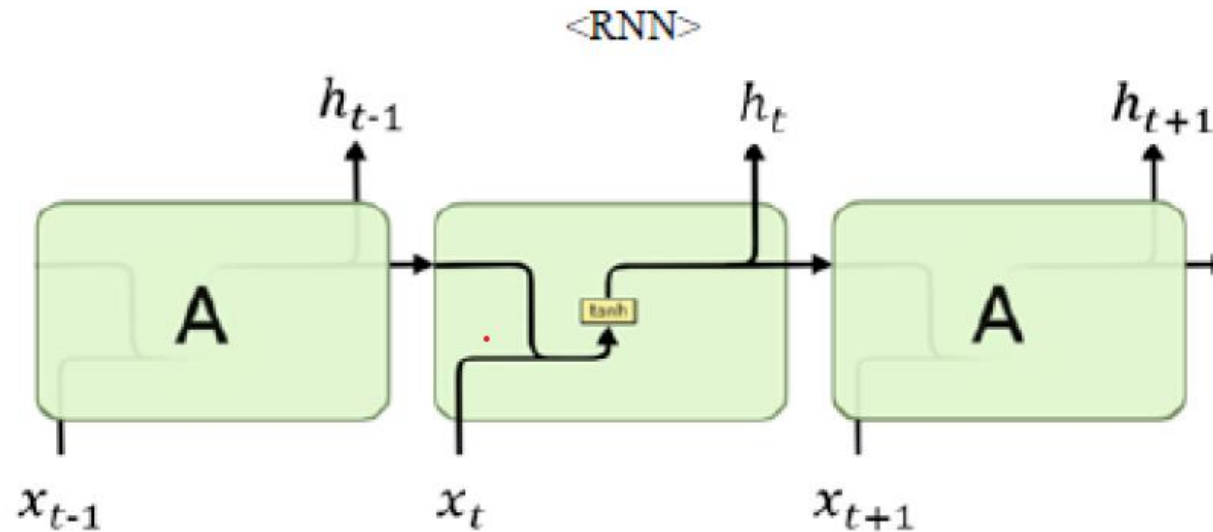
# Deep Learning



Note: This figure exemplifies a deep learning network for image recognition. The network consists of an input layer, three hidden layers, and an output layer. Input layer consists of nodes (the blue circles) storing the RGB values extracted from each pixel of the

# Deep Learning - RNN/LSTM

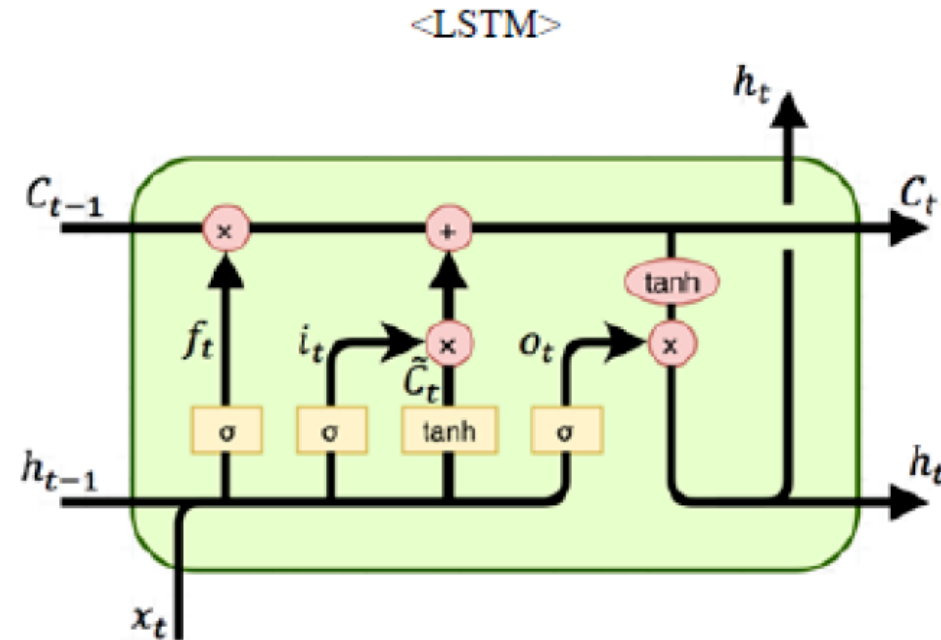
---



Note: RNN cell(square in the diagram) takes input  $x_t$  and hidden layer  $h_{t-1}$ (containing past information) and generates  $h_t$ . Output at time  $t$  is calculated by a linear combination of nodes in  $h_t$

Source: Amidi (2021)

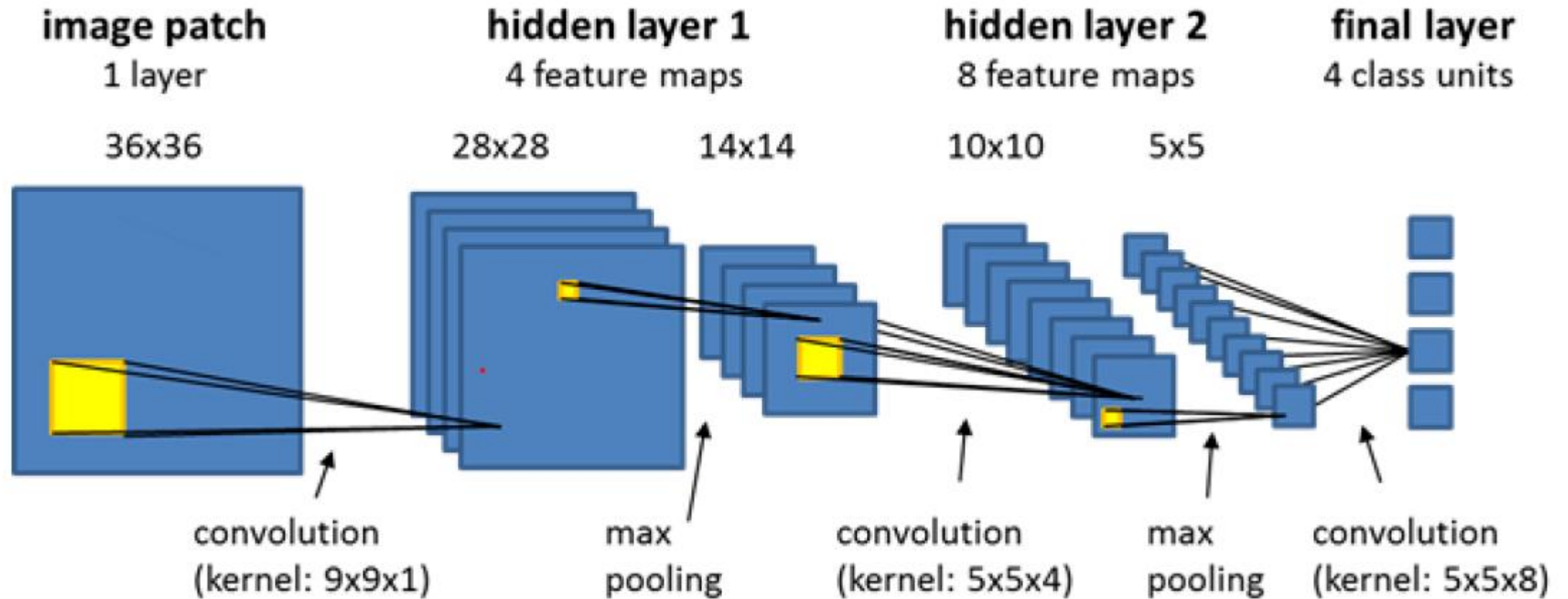
# Deep Learning - RNN/LSTM



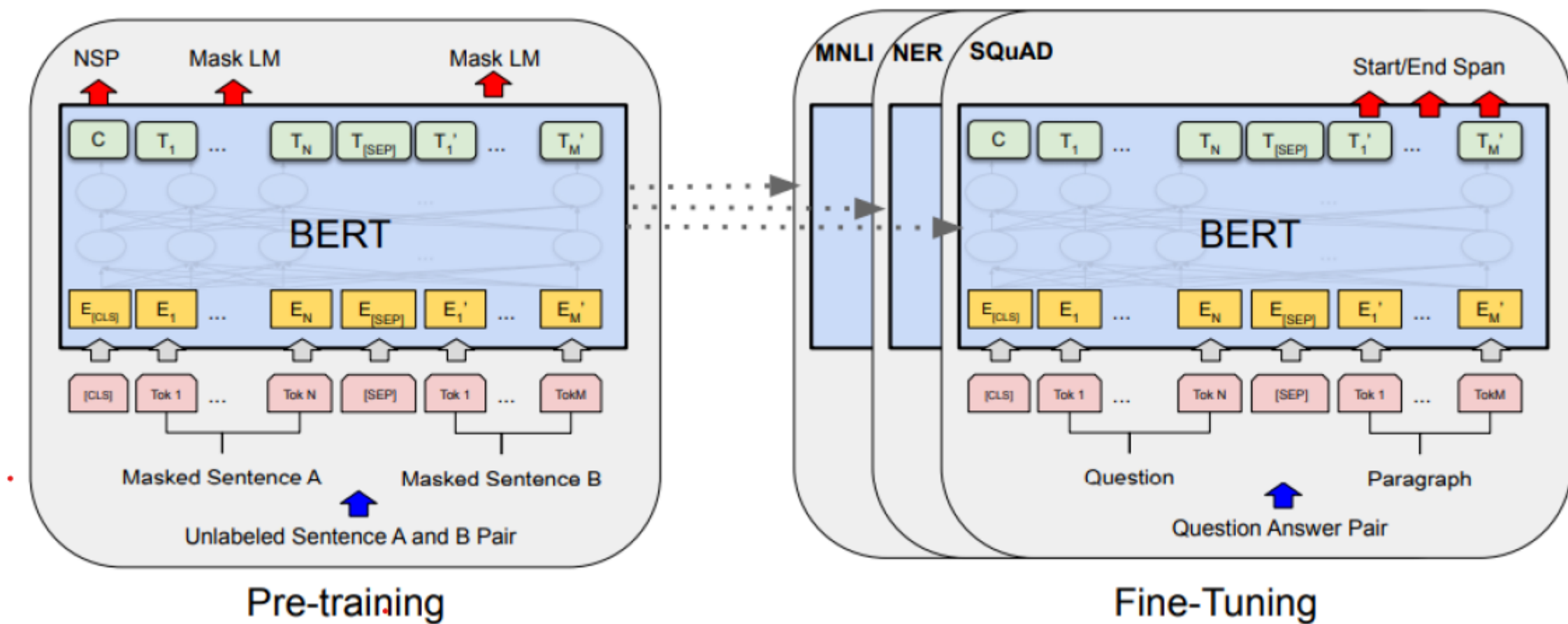
Note: LSTM cell (square in the diagram) takes input  $x_t$  and hidden layer  $h_{t-1}$  and generates hidden state  $h_t$  and cell state  $C_t$  at time  $t$ . Long-term information of LSTM cell at time  $t$  is passed on to the next LSTM cell by cell state  $C_t$ . Output at time  $t$  is calculated by a linear combination of  $h_t$ .

Source: Olah (2015)

# Deep Learning - CNN



# 인공지능 언어모델



**감사합니다**