클라우드 컴퓨팅과 AI서비스
(7주차)

융합학과 권오영

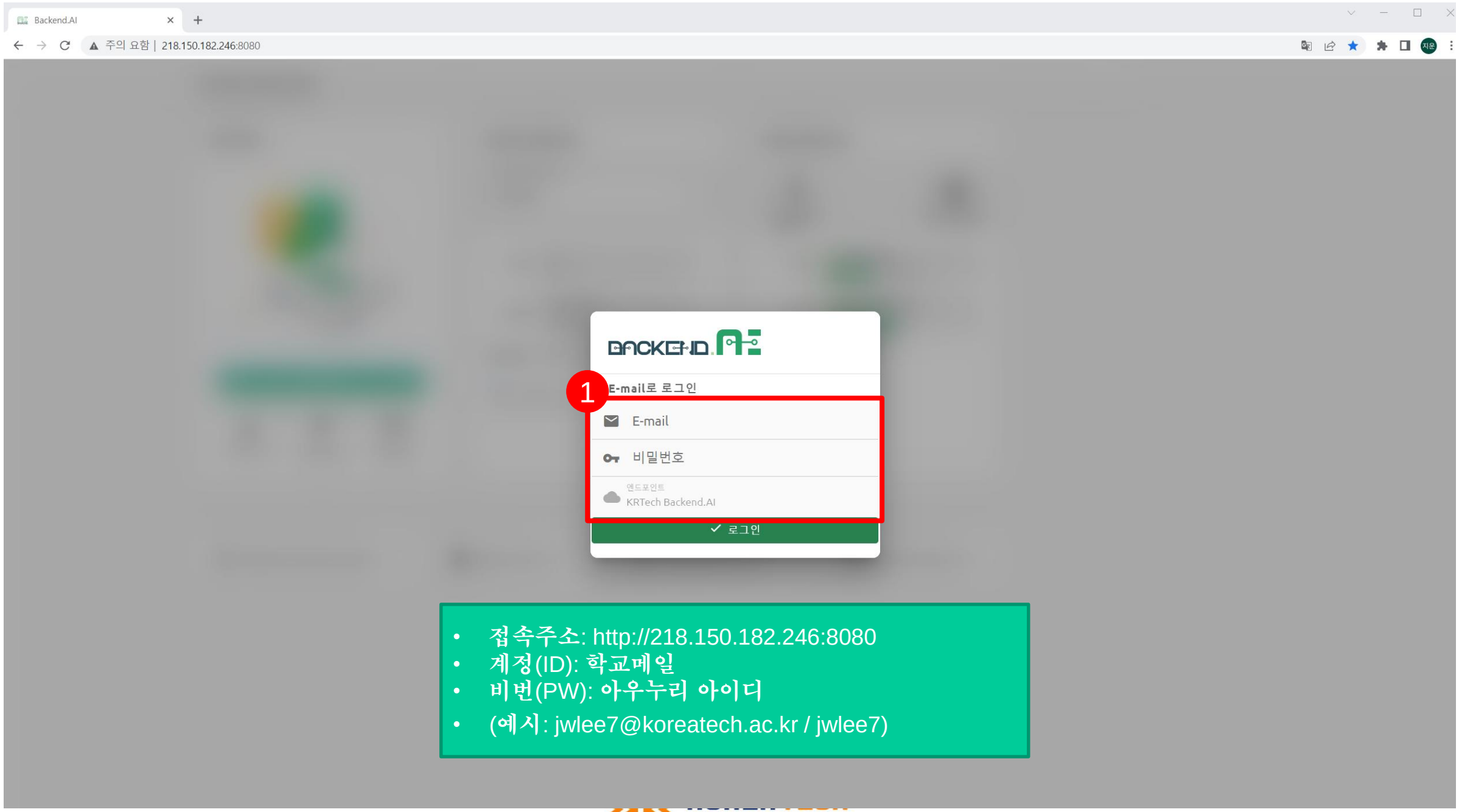
oykwon@koreatech.ac.kr

KOREATECH 한국기술교육대학교

학습내용

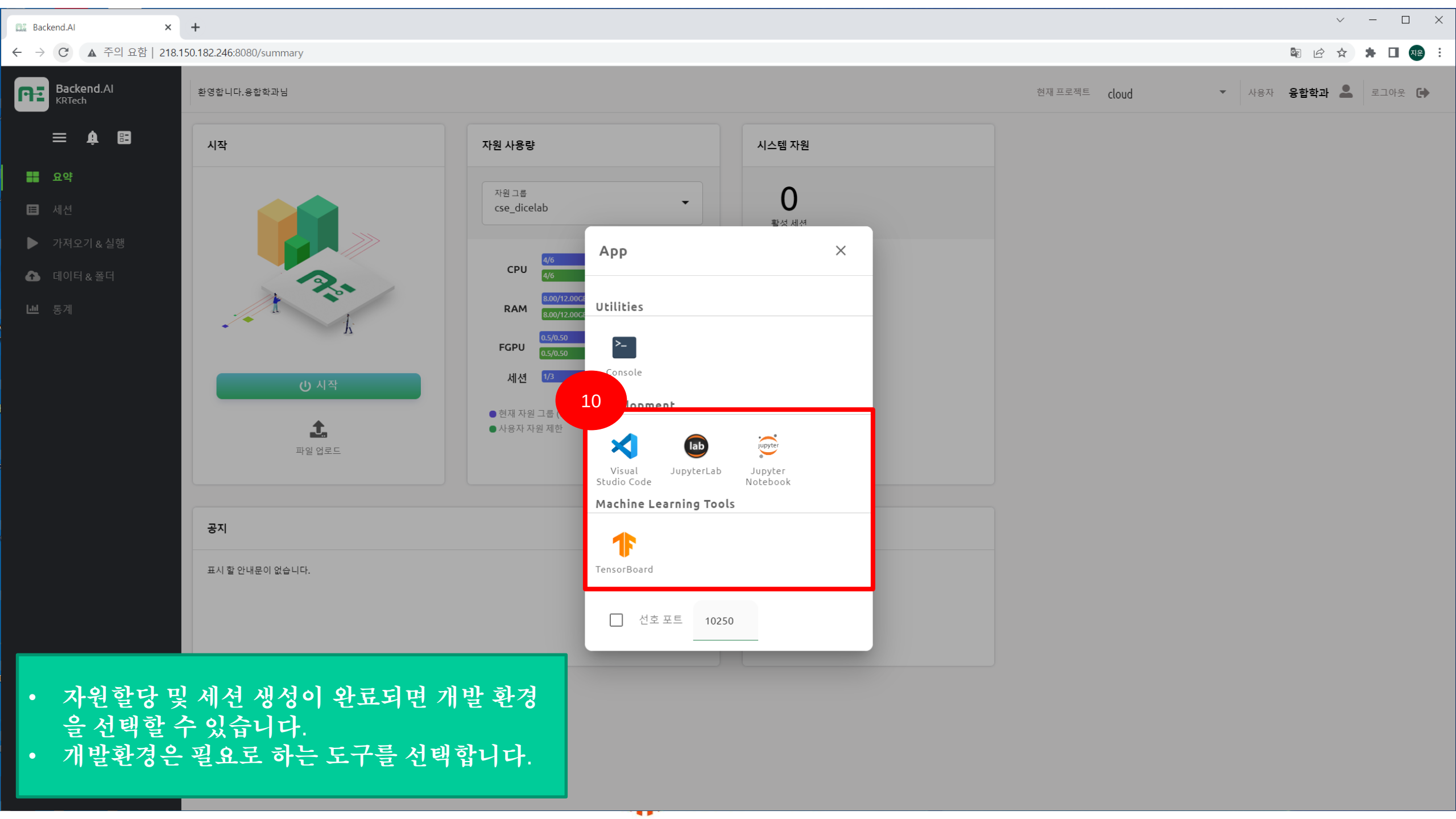
- 교내 클라우드 환경
- Markdown 응용(Note-taking APP)
- 인공지능/미래직업
- 기계학습

교내 GPU 클라우드



- 접속주소: http://218.150.182.246:8080
- 계정(ID): 학교메일
- 비번(PW): 아우누리 아이디
- (예시: jwlee7@koreatech.ac.kr / jwlee7)

한국기술교육대학교


Backend.AI
218.150.182.246:8080/summary
Backend.AI
KRTech
환영합니다.융합학과님
현재 프로젝트
cloud
사용자
융합학과
로그아웃
요약
세션
가져오기 & 실행
데이터 & 폴더
통계
시작
시작
파일 업로드
자원 사용량
자원 그룹
cse_dicelab
CPU
4/6
4/6
RAM
8.00/12.00GB
8.00/12.00GB
FGPU
0.5/0.50
0.5/0.50
세션
1/3
현재 자원 그룹
사용자 자원 제한
시스템 자원
0
활성 세션
공지
표시 할 안내문이 없습니다.
App
Utilities
Console
10
Visual Studio Code
JupyterLab
Jupyter Notebook
Machine Learning Tools
TensorBoard
선호 포트
10250
• 자원할당 및 세션 생성이 완료되면 개발 환경을 선택할 수 있습니다.
• 개발환경은 필요로 하는 도구를 선택합니다.

Analyses about the COVID-19 virus (예제)

❖ 사이트 정보 https://github.com/twiecki/covid19

❖ Docker, 주피터 등을 활용할 수 있는 좋은 예제

❖ 교내 클라우드에서 수행

- covid-19 맵 응용을 교내클라우드에서 수행해보기
  https://opensource.com/article/20/4/python-map-covid-19
- Colab 코드를 교내 클라우드에서 수행해보기
- 골프자세교정AI만들기
  (https://www.youtube.com/watch?v=PeuE0nLQqzU)
- **Algorithmic Trading Strategy Using Python**
  (https://www.youtube.com/watch?v=SEQbb8w7VTw)
- **A Machine Learning Stock Trading Strategy Using Python**
  (https://www.youtube.com/watch?v=Whj0u6T6nBg)

마크업 언어(MARKUP LANGUAGES)

마크업 언어(markup language)

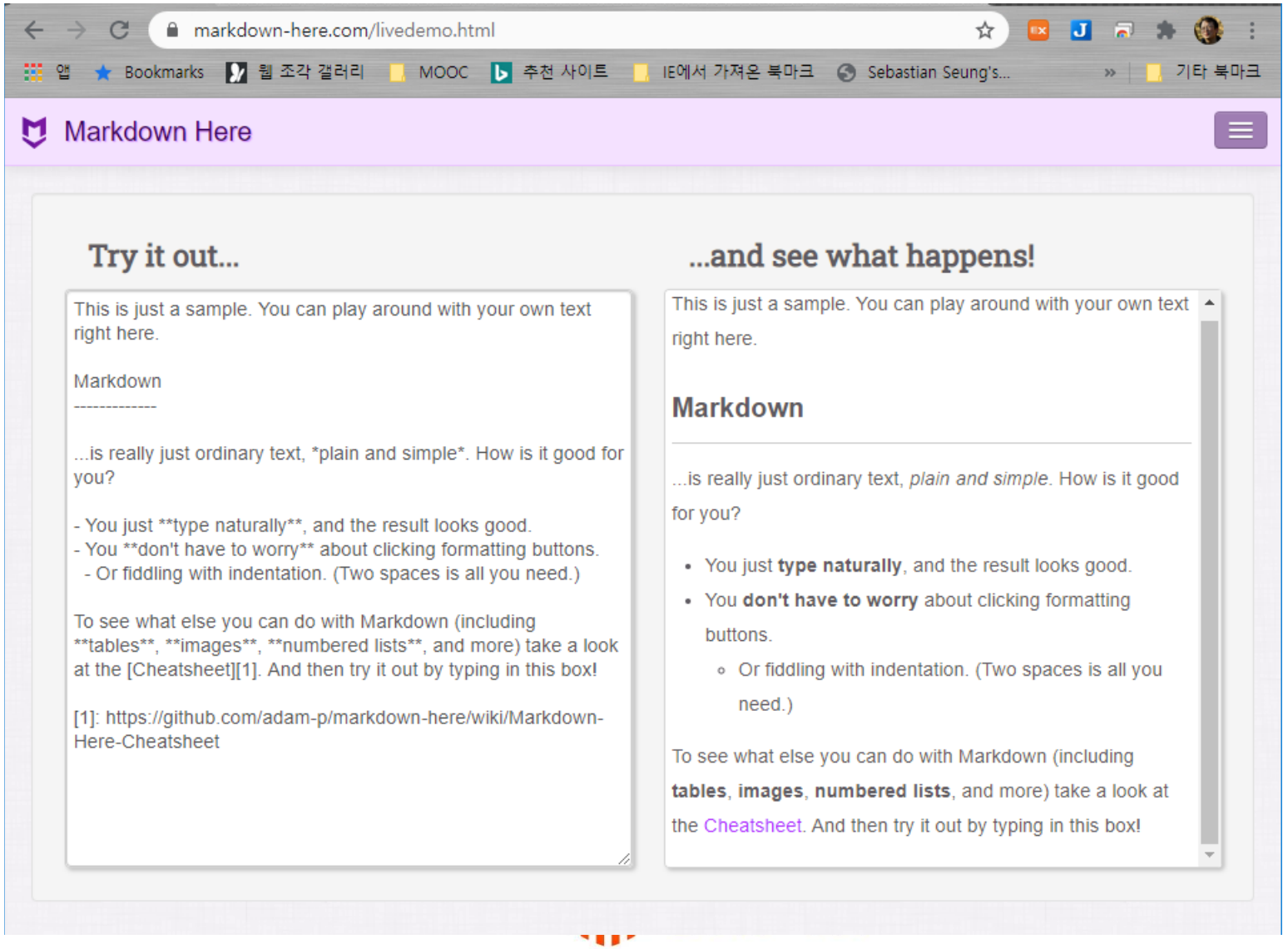
❖ **마크업 언어**(markup 言語, markup language)는 태그(mark) 등을 이용하여 데이터의 구조를 명기하는 언어의 한 가지이다. (위키정의)

❖ Markdown 언어

- http://nolboo.github.io/blog/2014/04/15/how-to-use-markdown/
- 온라인상에 메시지를 전달하기 위한 readme 파일을 작성하는데 많이 이용된다. 예로 github 사이트의 readme.md 파일
- 2004년 John Gruber가 제안 (http://daringfireball.net/projects/markdown/)
- Html보다 간단하고, md파일을 html파일로 변경하는 툴이 다수 존재한다.
- http://daringfireball.net/projects/markdown/syntax
  - ✓ 제안자가 문법을 자세히 설명해 놓은 곳
- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
  - ✓ Cheatsheet의 내용만 살펴보아도 쉽게 문서를 작성할 수 있다.

Markdown 문서

http://markdown-here.com/livedemo.html



Opensource.com: Markdown Cheat Sheet

BY MATT BROBERG

Markdown is the most widely adopted way to format documents using plain text syntax. Here are the essentials.

STYLIZE TEXT

italic	"italic"
bold	"**bold**"
~~strikethrough~~	~~strikethrough~~
`code in-line`	\`code in-line\`
code block	\`\`\`code block\`\`\`
> text	blockquote

HEADERS AND BREAKS

# Text	H1
## Text	H2
### Text	H3
---	Horizontal Rule

LISTS

Lists can be unordered or ordered, which is determined by the symbol used to list them. Unordered lists have either `-`, `+`, or `*`, while ordered lists require any integer followed by a `.` or `)` character.

An ordered list:

```
1. First item
2. Second item
3. Third item
```

Unordered List:

```
- First item
- Second item
- Third item
```

Combining the two:

```
1. First ordered list item
1. Second ordered list item (which
   shows as 2.)
1. (Markdown ignores the written
   number in favor of enumerating
   by integer)
    - Indented lists require four
      spaces in most specifications
    - Sublists are implementation
      specific
    - Try it out and render to be
      sure it works
1. Continuing the list after an
   indentation
```

TABLES

Tables are a common extension of the official markdown definition (CommonMark). They can be formatted in quite a lot of ways:

```
| Column Title       | Another Column                | One More |
|:-------------------|:----------|:---------|
| **Bolded text**    | Columns don't need to be aligned  |          |
| Or even filled in |                                 |   code   |
```

Renders as:

Column Title	Another Column	One More
bolded text	Columns don't need to be aligned	
Or even filled in		`code`

That can be painful to look at. Here's a prettier example with alignment based on the placement of the `:` in the table structure:

```
| Tables   |      Are      | Cool |
|:---------|:-------------:|-----:|
| col 1 is | left-aligned  |   $1 |
| col 2 is |   centered    |   $2 |
| col 3 is | right-aligned |   $3 |
```

Tables	Are	Cool
col 1 is	left-aligned	$1
col 2 is	centered	$2
col 3 is	right-aligned	$3

Tables are always a little tricky to remember. Use available online tools to help build them, like: https://csvtomd.com

LINKS

There are two ways to create links. The most commonly used format includes brackets followed by parentheses: `[]()`. To render a link to an image, begin with a `!`.

Examples.

```
[Link inline this way](https://opensource.com)
[Or add a title for the link](https://opensource.com "Google's Homepage")
[Here is a relative link within a repository](../blob/master/LICENSE)
![I link to an image](path/to/image.png)
```

The second format involves brackets, `[][]` or `[]`, followed by a reference formatted with a bracket and colon, `[]:`, anywhere else in the document.

Examples.

```
[Reference-style link][Case-insensitive Reference Text]
[Numbers are commonly used][1]
[Text can link on its own as a reference]
[case-insensitive reference text]: https://opensource.com
[1]: https://opensource.com
[text can link on its own as a reference]: https://opensource.com
```

URLs, in and out of angle brackets, will automatically get turned into links with most implementations.

Example.

```
Both https://opensource.com and <https://opensource.com> will render as
links, as well as opensource.com on some renderings.
```


Opensource.com: Markdown Cheat Sheet BY MATT BROBERG

BEST OF GITHUB FLAVORED MARKDOWN

Task lists are fantastic usage of the GitHub-specific implementation:

```
Task List
- [x] Step one is complete
- [ ] Step two in this unordered list is
- [x] Step three is done as well
```

Renders as:

Task List

- [x] Step one is complete
- [ ] Step two in this unordered list is
- [x] Step three is done as well

Drop-downs are an incredible feature to tidy up files:

```
<details>
  <summary>Q1: What is the best website in
     the world? </summary>
  A1: Opensource.com
</details>
```

Renders as a clickable drop-down menu. See the example at github.com/opensourceway/markdown-example

You can also have language-specific syntax highlighting. Instead of having a code block of black-and-white text, append the language to the first set of backticks to have highlighting enabled:

```
<html>
<head>
  <meta content="text/html;charset=utf-8"
   http-equiv="Content-Type" />
</head>
<body>
  <script src='./pkg/my_wasm_library.js'></script>
  <script>
    window.addEventListener('load', async () => {
      // Load the wasm file
      await wasm_bindgen('./pkg/my_wasm_library_bg.wasm');
      // Once it's loaded the `wasm_bindgen` object is
      // populated with the functions defined in our Rust code
      const greeting = wasm_bindgen.excited_greeting("Matt")
      console.log(greeting)
    });
  </script>
</body>
</html>
```

Nearly all programming languages are supported using this syntax (python, ruby, go, rust, javascript, and java to name a few). See GitHub's documentation for the full list: https://help.github.com/en/articles/creating-and-highlighting-code-blocks#syntax-highlighting

GITLAB SPECIFIC REFERENCES

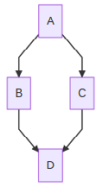
GitLab, the second largest Git-based repository on the Internet, has unique global references designed for teamwork.

@user_name	specific user
@group_name	specific group
@all	entire team
#123	issue
!123	merge request
$123	snippet
~123	label by ID
~bug	one-word label by name
9ba12248	specific commit
9ba12248...b19a04f5	commit range comparison
`[README](doc/Readme)`	repository file references
`/tableflip <comment>`	Quick reaction that includes (ノ°□°)ノ ⌒ ┻┻

You can also design flow diagrams:

```
mermaid
graph TD;
  A-->B;
  A-->C;
  B-->D;
  C-->D;
```

Becomes:



THE BEST OF BOTH GITHUB AND GITLAB

Emojis bring both formats together. Use everything from `:abc:` to `:zap:` to add emojis to your markdown 🎉.

A searchable list of emoji icons is available at https://www.webfx.com/tools/emoji-cheat-sheet/

REFERENCES

https://commonmark.org/

https://spec.commonmark.org/0.28/

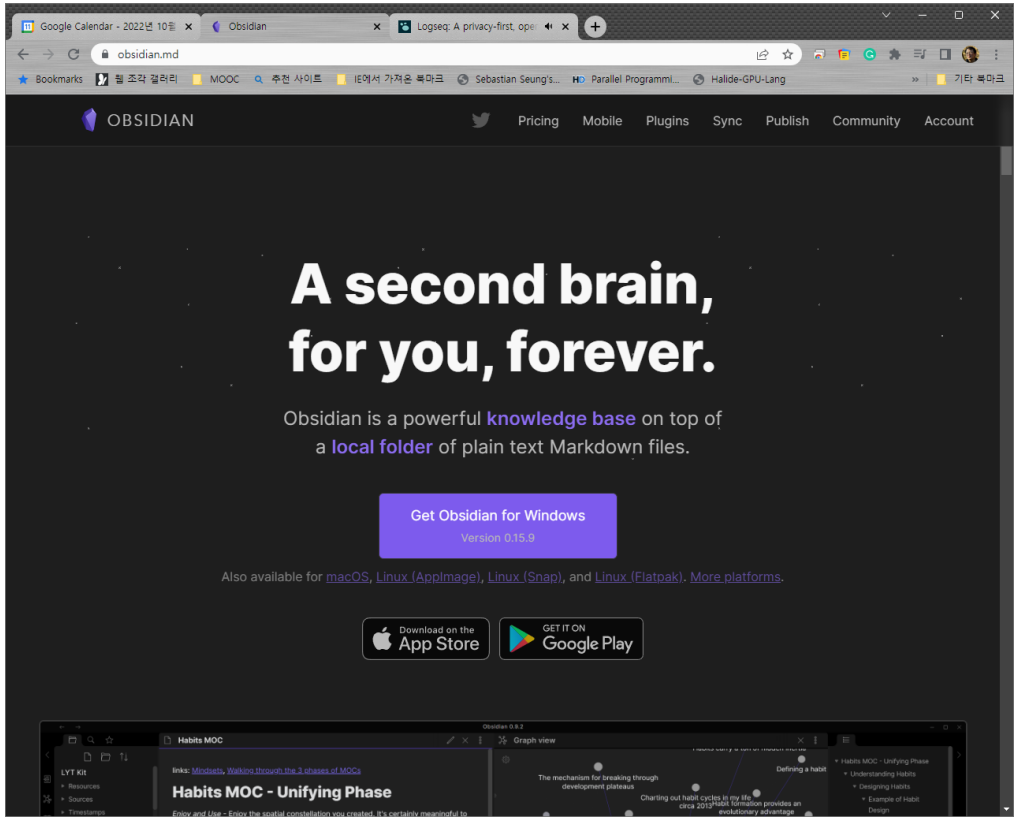
https://github.github.com/gfm/

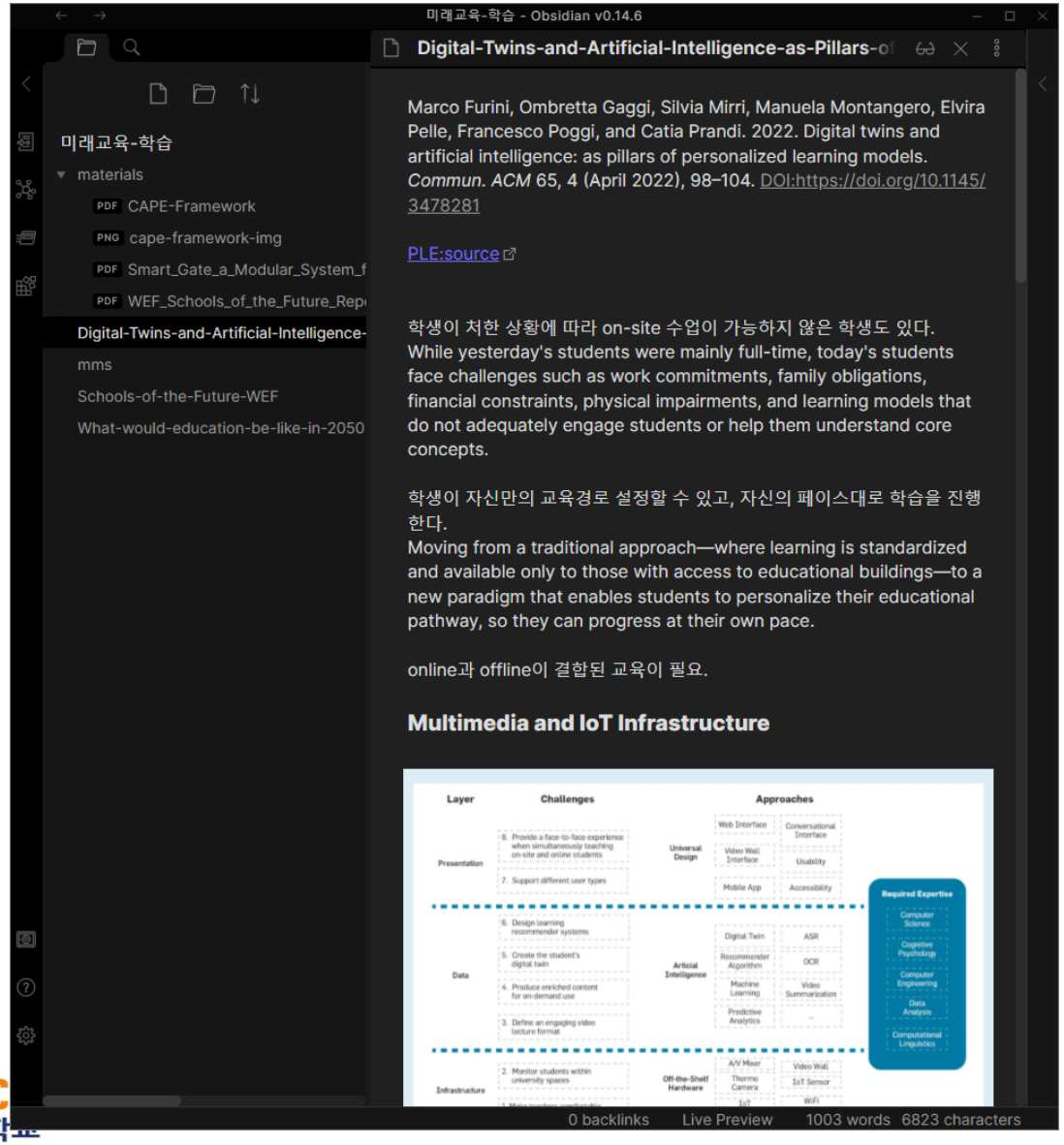
https://docs.gitlab.com/ee/user/markdown.html

opensource.com Twitter @opensourceway | facebook.com/opensourceway | CC BY-SA 4.0

마크다운언어 응용 (Note-taking APP)

- Obsidian (https://obsidian.md/)
  - https://github.com/ieshreya/Obsidian-Cheat-Sheet

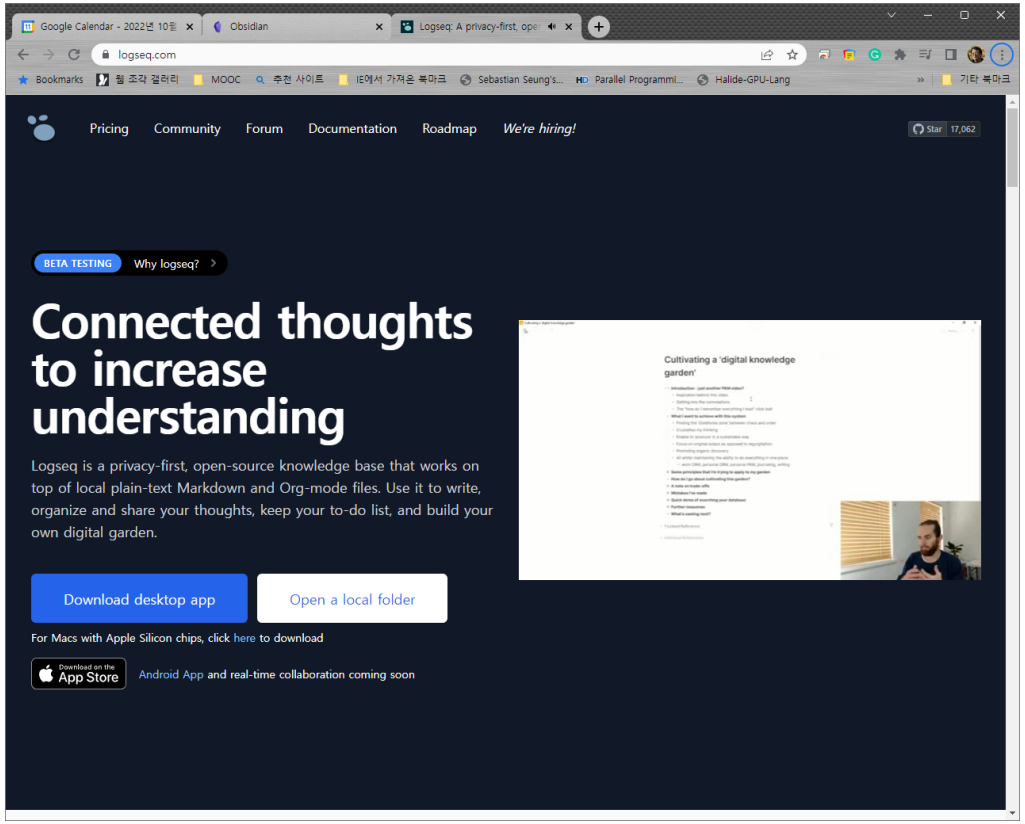


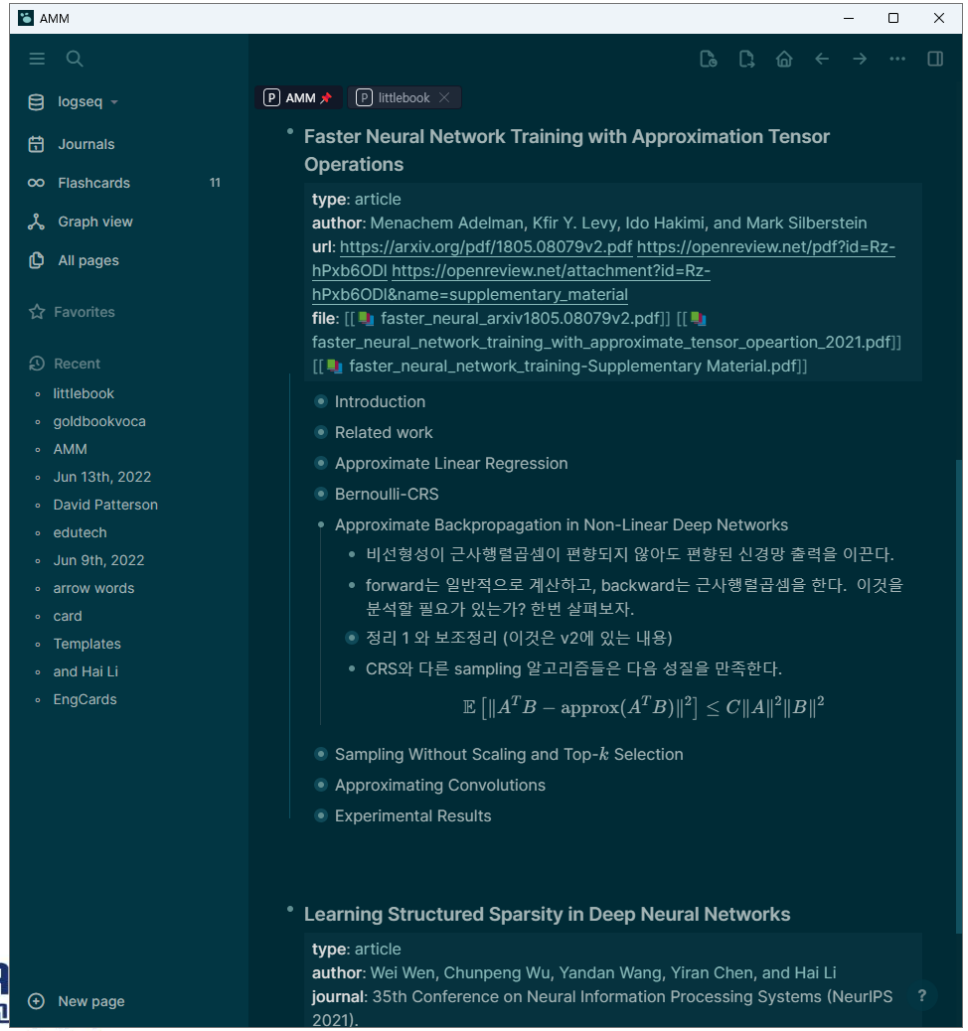


KOREATECH 한국기술교육대학교

마크다운언어 응용(Note-taking APP)

- Logseq (https://logseq.com/)
  - https://cheatography.com/bgrolleman/cheat-sheets/logseq/





인공지능/미래직업

인공지능

- ❖ 인공지능 : 인간의 학습능력, 추론능력, 지각능력, 언어이해능력등을 컴퓨터 프로그램으로 구현한 기술
- ❖ 위키의 정의(https://ko.wikipedia.org/wiki/인공지능)
  - ▪ 초기 인공지능 연구에 대한 대표적인 정의는 다트머스 회의에서 존 매카시가 제안한 것으로 "**기계를 인간 행동의 지식에서와 같이 행동하게 만드는 것**"이다.
  - ▪ **강인공지능(범용인공지능) :** 어떤 문제를 실제로 사고하고 해결할 수 있는 컴퓨터 기반의 인공적인 지능을 만들어 내는 것에 관한 연구다.
    - ✓ 인간의 사고와 같이 컴퓨터 프로그램이 행동하고 사고하는 인간형 인공지능.
    - ✓ 인간과 다른 형태의 지각과 사고 추론을 발전시키는 컴퓨터 프로그램인 비인간형 인공지능.
  - ▪ **약인공지능(weak AI) :** 어떤 문제를 실제로 사고하거나 해결할 수는 어떤 면에서 보면 지능적인 행동으로 볼 수 있는 인공지능
    - ✓ 주로 미리 정의된 규칙의 모음을 이용해서 지능을 흉내내는 컴퓨터 프로그램을 개발

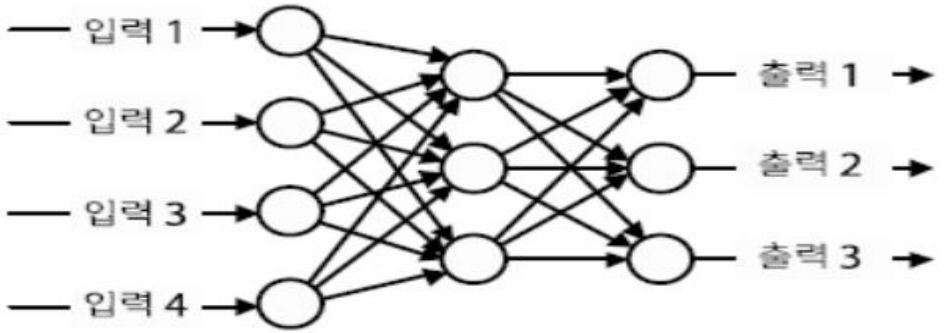
인공신경망

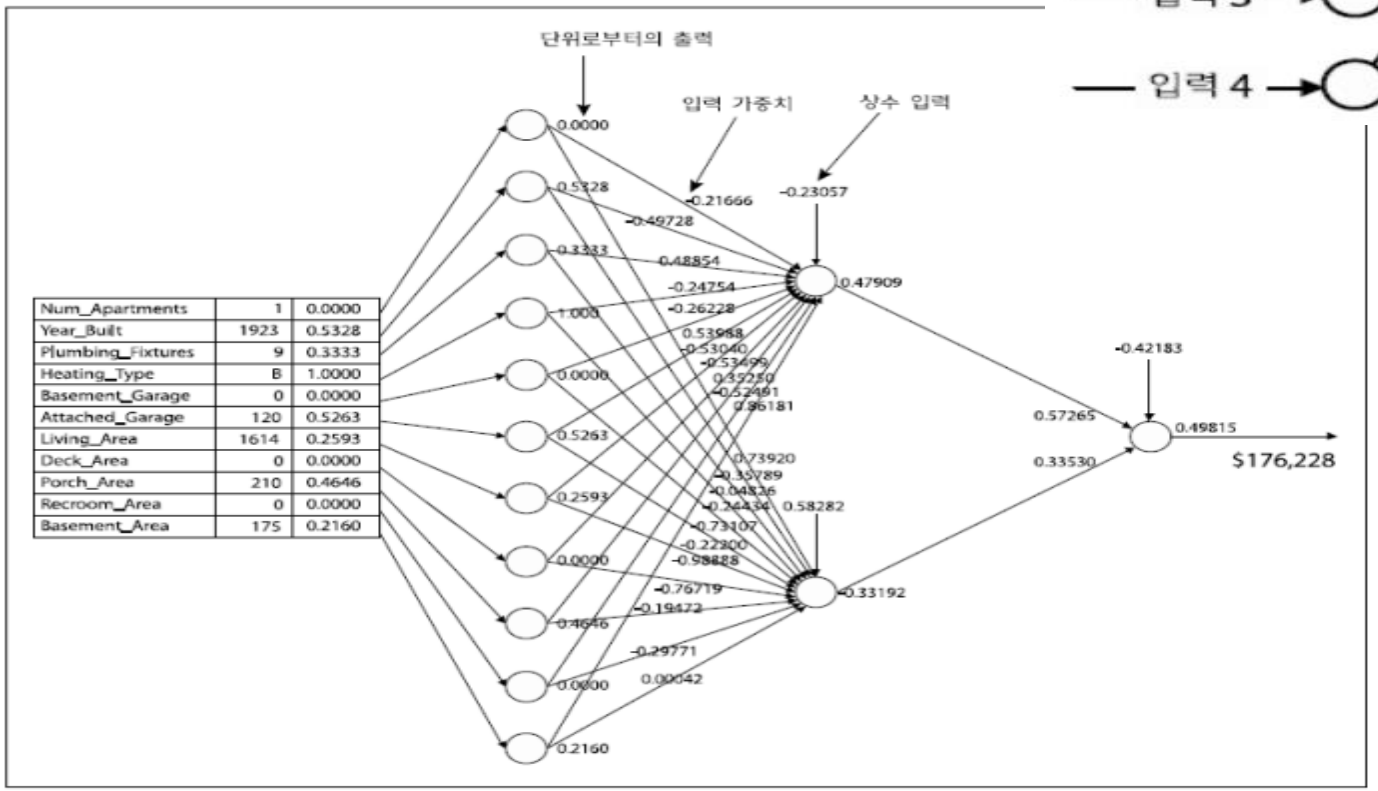
- 인간의 신경세포를 흉내낸 시스템
- 1943년 Warren McCullouch와 Walter Pits
  - 예일 대학교 신경심리학자, 논리학자
  - 생물학적인 뉴런의 작동에 설명하기 위한 간단한 모형을 제안
- 1950년대, 공학자들은 맥컬로크와 피츠의 연구에 기초한 퍼셉트론(perceptrons) 모형을 사용
- 1968년 MIT 교수 Papert와 Minsky
  - 단순한 신경망의 이론적 한계 지적
  - 1970년대까지 침체기
- 1982년 캘리포리아공대 John Hopfield
  - 역전파 알고리즘(Back propagation)
  - 이전 방법의 이론적 함정들을 극복
  - 인공신경망의 르네상스

KOREATECH 한국기술교육대학교

인공신경망

❖ 다층네트워크





출처: 김종우, 김선태, 경영을 위한 데이터마이닝, 한경사, 2009

KOREATECH 한국기술교육대학교

인공신경망

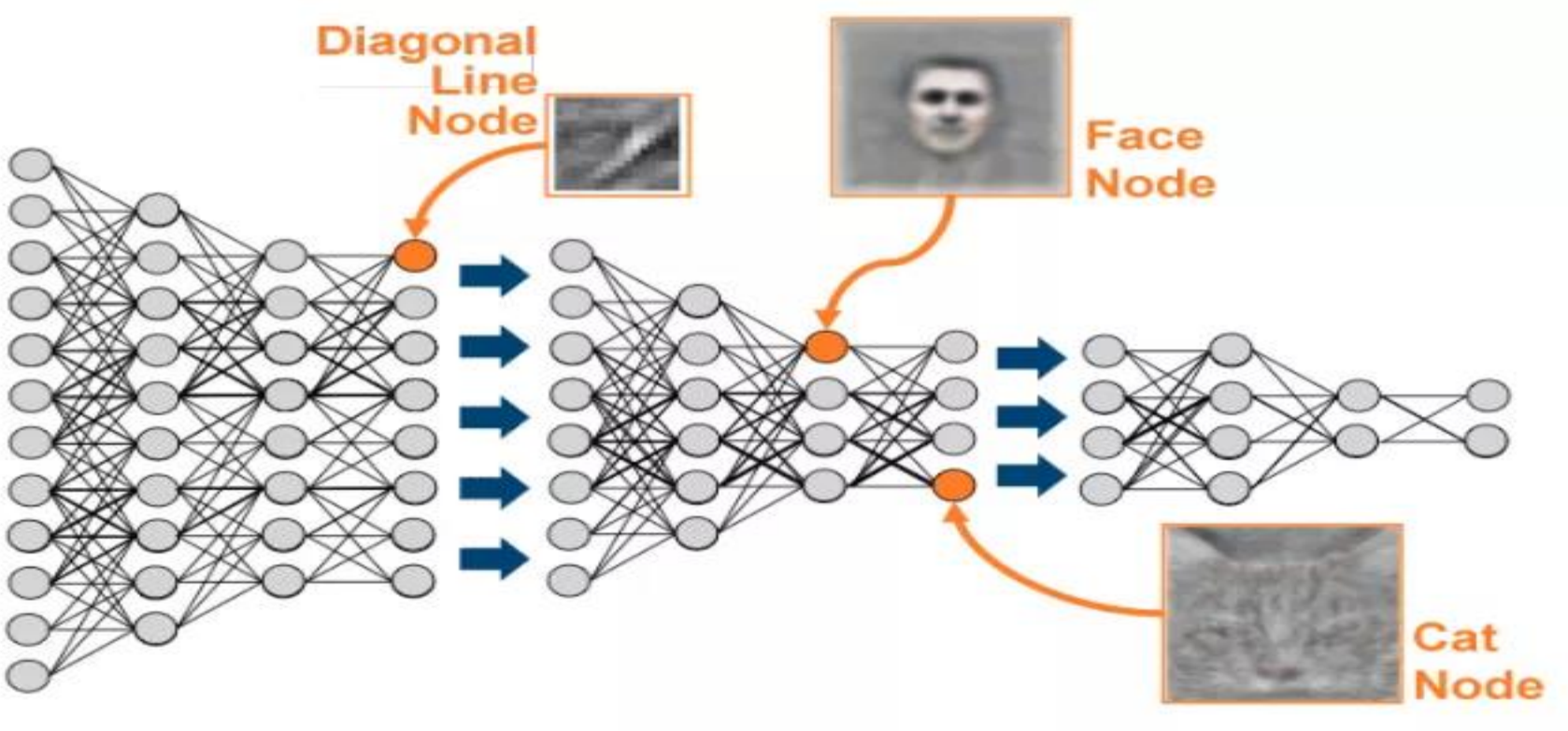
- 1980년대 인공신경망에 대한 연구는 연구실에서 상업계로 전이
  - 사기성 신용카드 거래 인식
  - 수표 금액 인식
  - 영상처리, 인공지능, 제어 등 다양한 분야에 적용
- 1990년대에 이르러 그 연구가 포화 상태에 이르고, 이내 한계가 보이기 시작하더니 곧 암흑기를 만남
- 2006년 Geoffrey Hinton
  - 데이터의 전처리과정(pre-training)을 통해 local minima에 빠지는 문제를 해결
  - “A fast learning algorithm for deep belief nets” 라는 논문
  - 인공신경망의 각 층을 먼저 비지도 학습방법(unsupervised learning)을 통해 잘 손질해주고, 그렇게 전처리한 데이터를 여러 층 쌓아올려 인공신경망 최적화를 수행하면 ‘이 산이 아닌가봐?’ 없이 훌륭한 결과를 만들어 낼 수 있다는 것을 보임
  - Deep Learning

KOREATECH 한국기술교육대학교

인공신경망

❖ 구글의 고양이 인식 (deep learning)

- 구글은 2012년 1,000대의 컴퓨터로 1,000만 개의 유튜브 이미지를 딥 러닝으로 분석해 사람과 고양이를 구분해 냈다.



KOREATECH
한국기술교육대학교

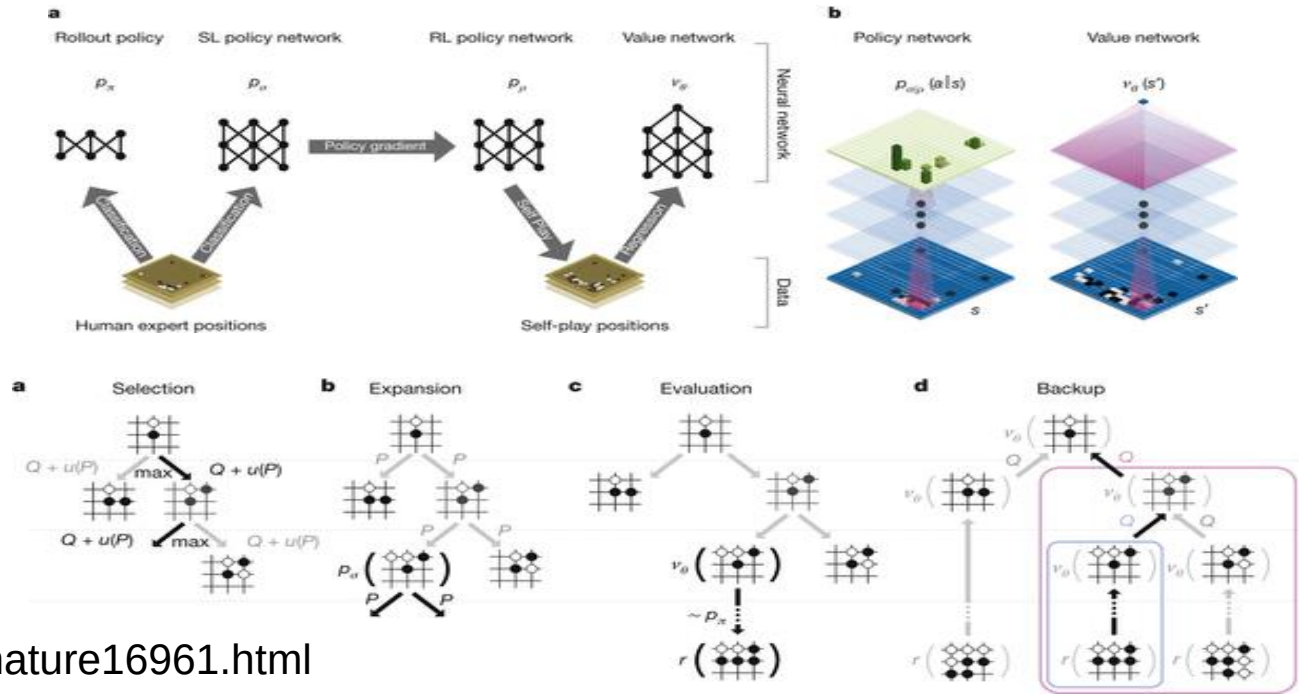
인공지능

- 1997년 IBM의 딥 블루 컴퓨터
  - 인간 체스 챔피언을 3.5-2.5로 승리한 인공지능을 채택한 컴퓨터
  - 가능한 모든 경우를 조사하여 다음 수를 결정하기 때문에 엄청난 계산처리 능력이 필요
    - ✓ 딥 블루는 30개의 노드로 구성된 컴퓨터이고 특별히 설계한 480개의 VLSI 체스 칩 장착 (1초당 200,000,000 개의 위치를 계산)
- 2011년 미국 퀴즈쇼 '제퍼디'에서 우승한 인공지능 컴퓨터 '왓슨'
  - 전 세계 백과사전과 위키피디아, 뉴욕타임스 아카이브, 성경 등 2억 페이지에 달하는 정보를 스스로 자기학습하며 수십만 가지의 질문에 답을 내놓음
  - '왓슨'은 2013년 미국 뉴욕의 MSKCC병원에서 암 환자를 위한 맞춤형 치료에 활용
    - ✓ 의학문헌 검토, 진료기록분석등을 수행 암 진단 보조
    - ✓ 유방암진단 정확도 91%이상 (전문의 초기 오진비율 20%~44%)
- 수많은 판례를 정보로 처리해 개별 사례에 맞는 법률적 조언을 제공하는 인공지능이 등장하기 시작

인공지능

❖ 알파고 (구글)

- 1202개의 CPU와 176개의 GPU 활용 (1378:1의 게임)
- Deep Neural Network, tree search 사용
- 프로바둑기사의 기보 16만건 확보해 3000만개 이상의 착점 학습
- 100만번(1000년)
  대국을 4주에 학습
- 타 인공바둑프로그램
  대비 99.8%의 승리
- 2016년 1월
  유럽 챔피언 판후이
  2단에게 5:0 승리



http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

KOREATECH 한국기술교육대학교

인공지능

❖ 인공지능의 발전 단계 (미국경제학자 타일러 코웬 교수의 예측)
인간보다열등 → 인간과 동등 → 인간을 보조 → 인간을 대체(?)

- 체스가 좋은 예임
  - ✓ 1997년 인간이 패배
  - ✓ 2000년대 프리스타일 가능 (인간고수, 인공지능, 인간+인공지능)
  - ✓ 2010년대 인공지능들간의 대전

❖ MIT의 브린욜프슨과 맥아피 교수 ‘ 2차 기계 시대’ 선언

- 1차 기계 시대: 대량 생산 기계가 단순 육체 노동을 대체한 시대
- 2차 기계 시대: 로봇과 인공지능이 복잡한 육체노동, 나아가 지식 노동마제 대체하는 시대

https://www.youtube.com/watch?v=EK2iSPjryRM
(Humans need not apply.)

미래의 직업

- 일이 비정형적이고, 이동성, 인지-조작 협응 능력, 판단/창의력, 감정적 대인 스킬 등이 중요할 수로 기계의 인간 대체 가능성은 낮아짐

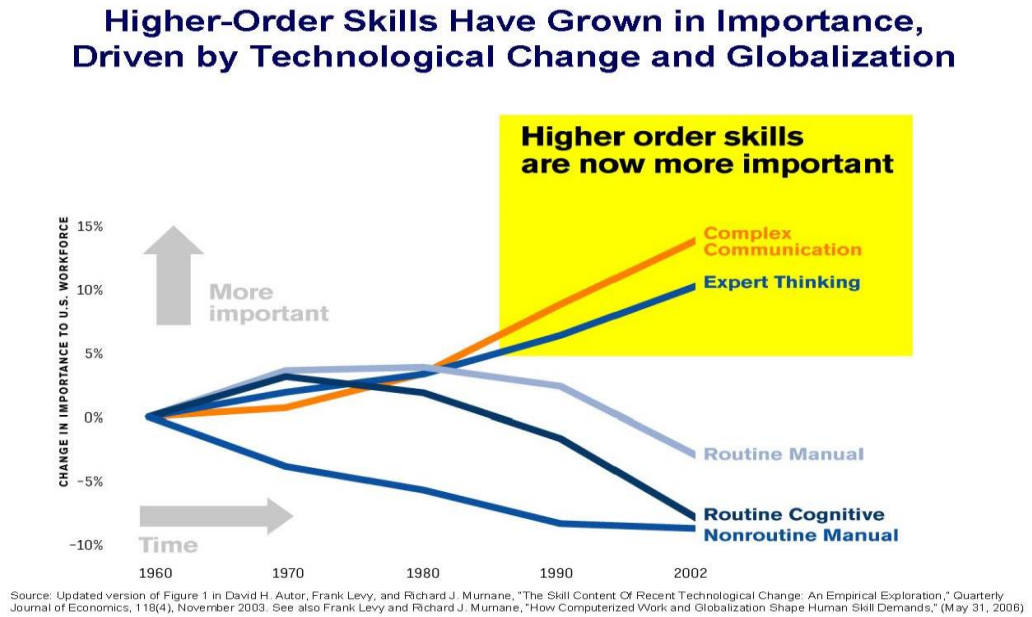
- 로봇, 인공지능의 도입으로 일자리가 없어지기도 하지만 새로운 일자리가 창출될 것임
  - 로봇, 인공지능을 설계, 관리, 지원하는 분야의 일자리 창출

- 인간과 기계의 협업
  - 기계: 방대한 정보 처리량, 빠른 연산력, 인적 오류 부재
  - 인간: 감성과 창의성, 불확실성에 대한 유연한 대응
  - 시간이 많이 걸리는 반복적인 직무를 인공지능에게 맡기고 인간은 창조적 직무에 집중

참고자료: 나준후,"로봇인공지능의 발전이 중산층을 위협한다." LG Business Insight 2014.7.9

미래직업에 필요한 능력

- ❖ 소통과 사고 (complex communication and expert thinking)
- ❖ 암기와 체득
  산업사회에 적합한 능력
- ❖ 미래사회
  감성(공감력), 사회성,
  창의성, 열정, 협업력,
  통찰력, 적응력, 평생학습
- ❖ 평생학습
  - 10년 마다 직업 바꾸는 시대
  - Khan academy, EdX, Coursera, Udacity, KMOOC, e-koreatech

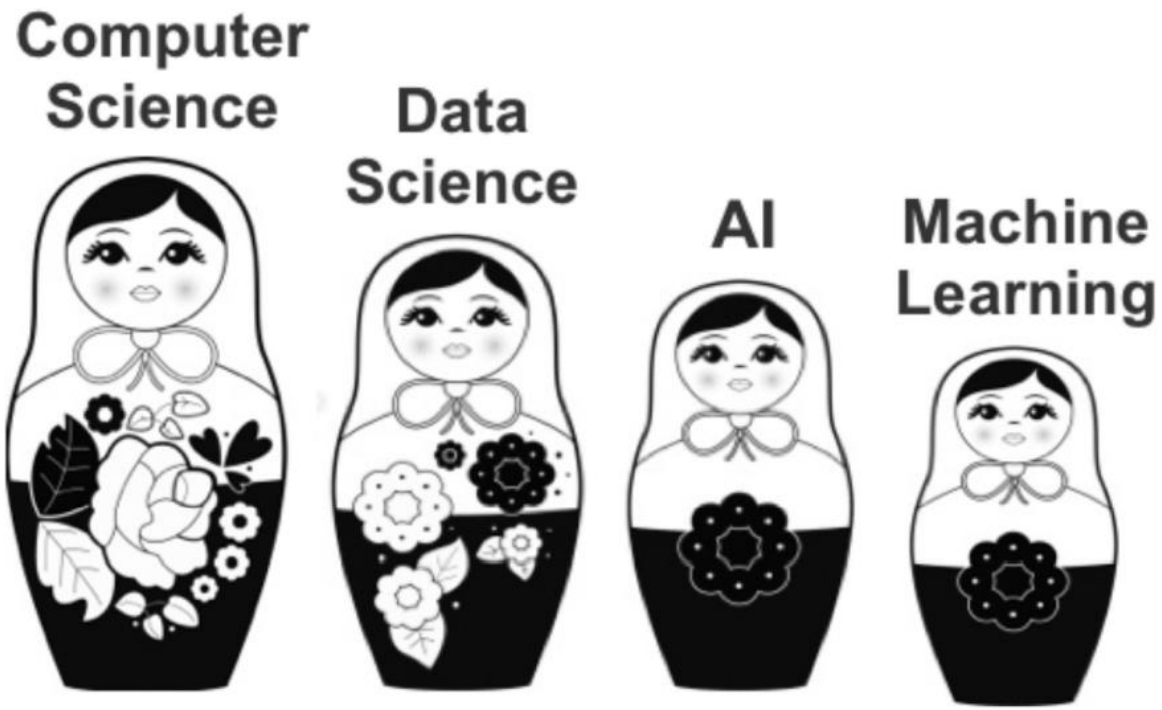


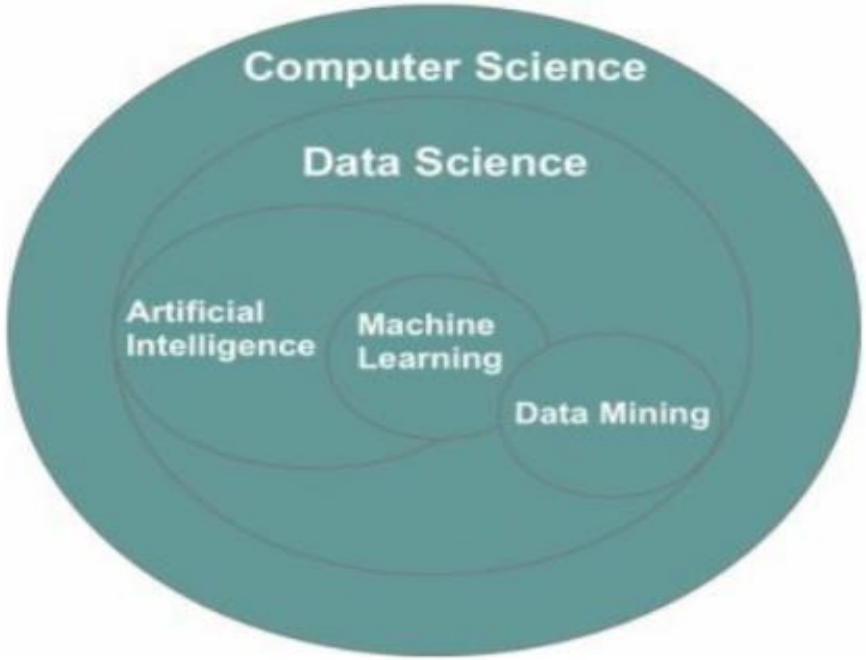
나음보단 다름이 필요 -> unique

KOREATECH 한국기술교육대학교

기계학습

The Anatomy of Machine Learning

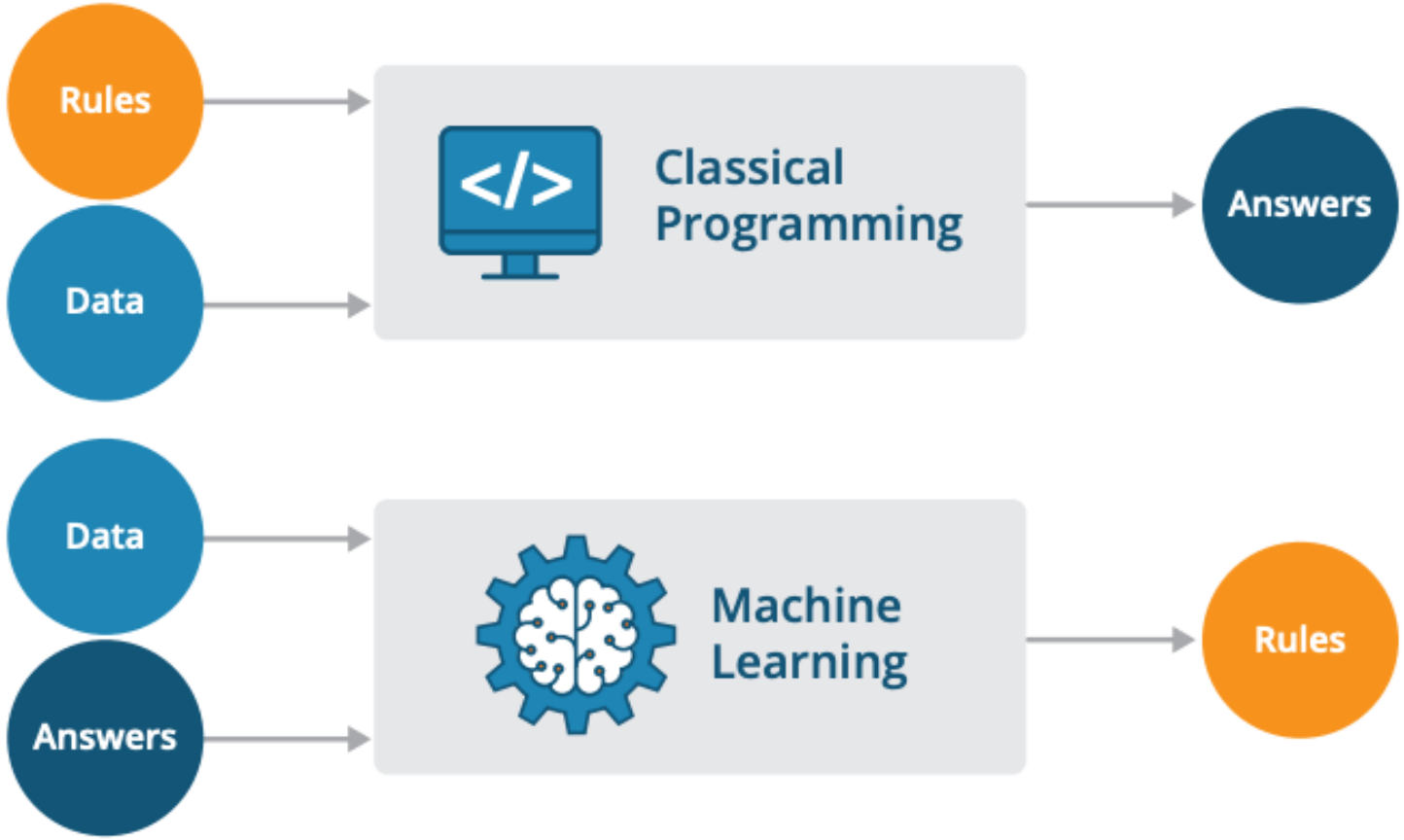




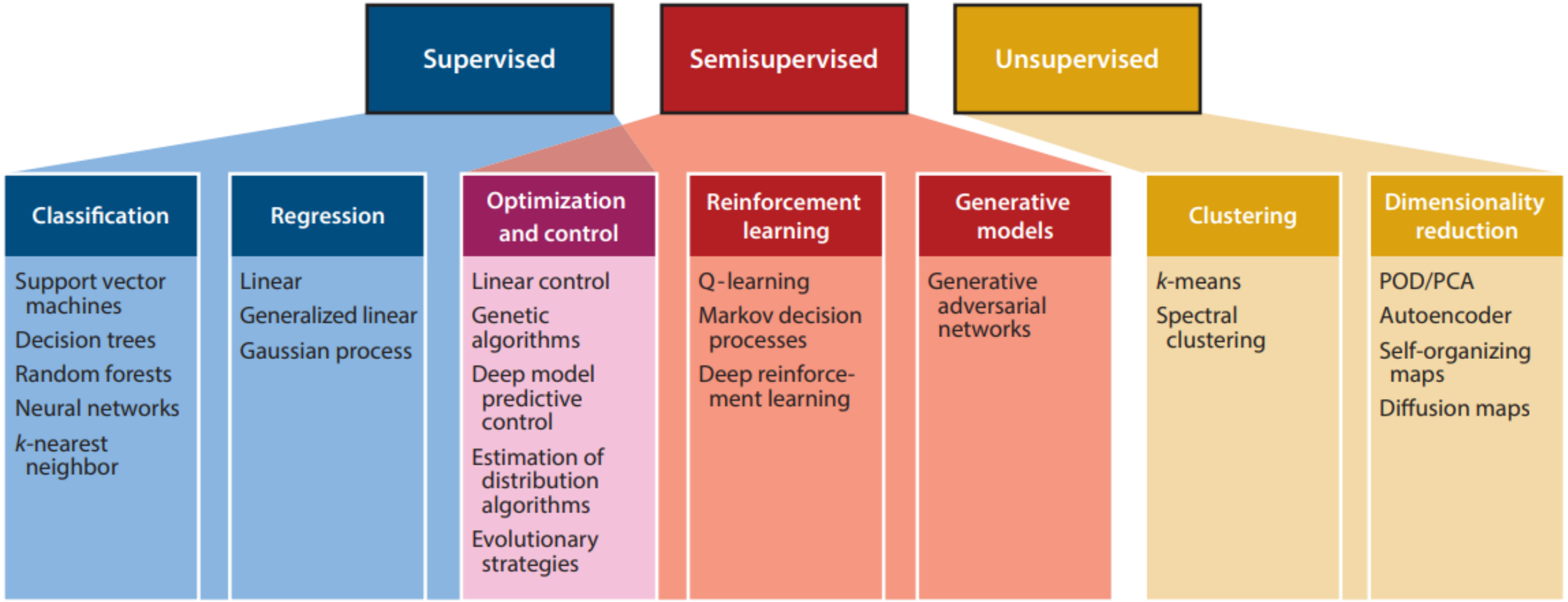
KOREATECH 한국기술교육대학교

Classical Programming vs ML

❖ Classical Programming vs ML

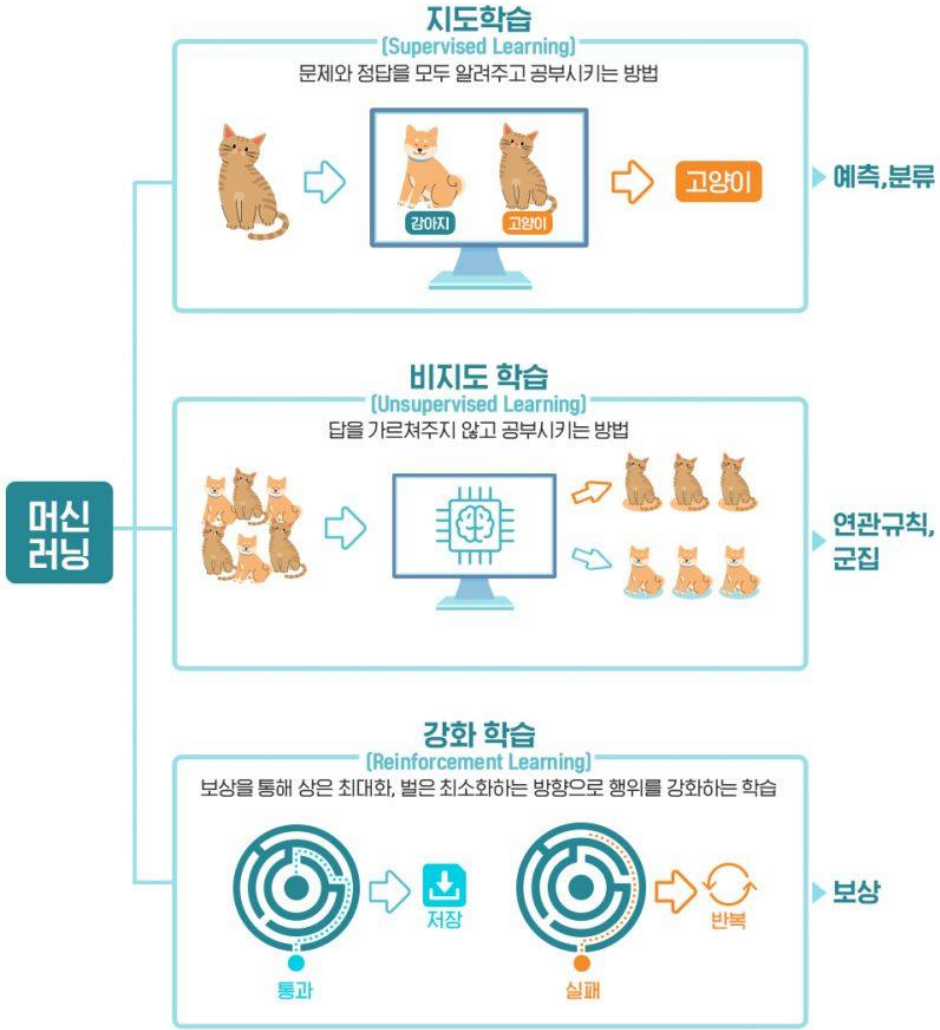


기계학습 분류

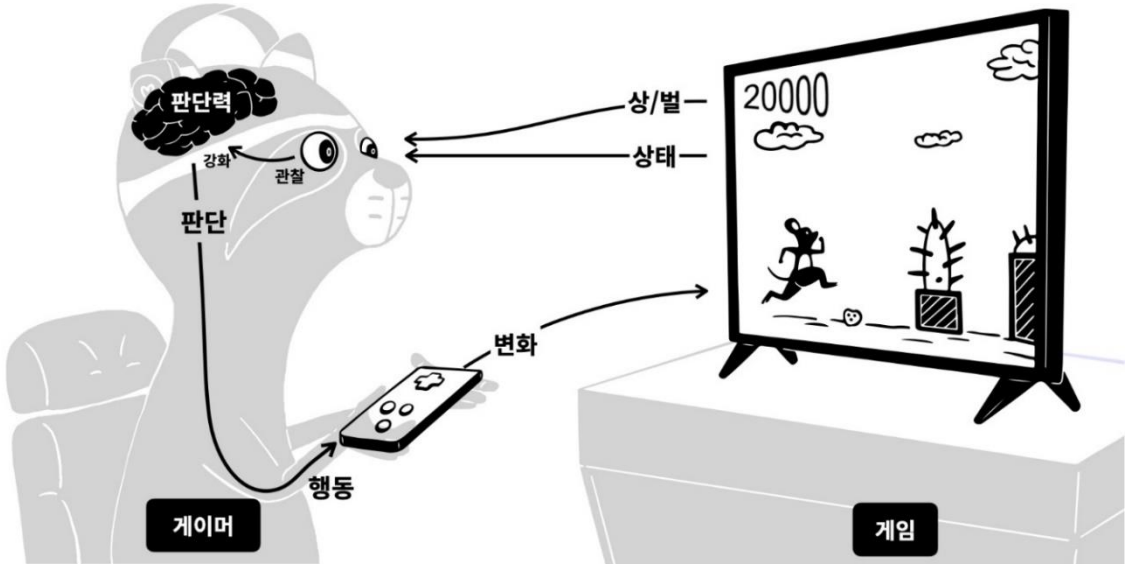


KOREATECH
한국기술교육대학교

기계학습방법



[출처] https://live.lge.co.kr/live_with_ai_01/



- 게임 ➡ 환경(environment)
- 게이머 ➡ 에이전트(agent)
- 게임화면 ➡ 상태(state)
- 게이머의 조작 ➡ 행동(action)
- 상과 벌 ➡ 보상(reward)
- 게이머의 판단력 ➡ 정책(policy)

[출처] https://opentutorials.org/course/4548/28949

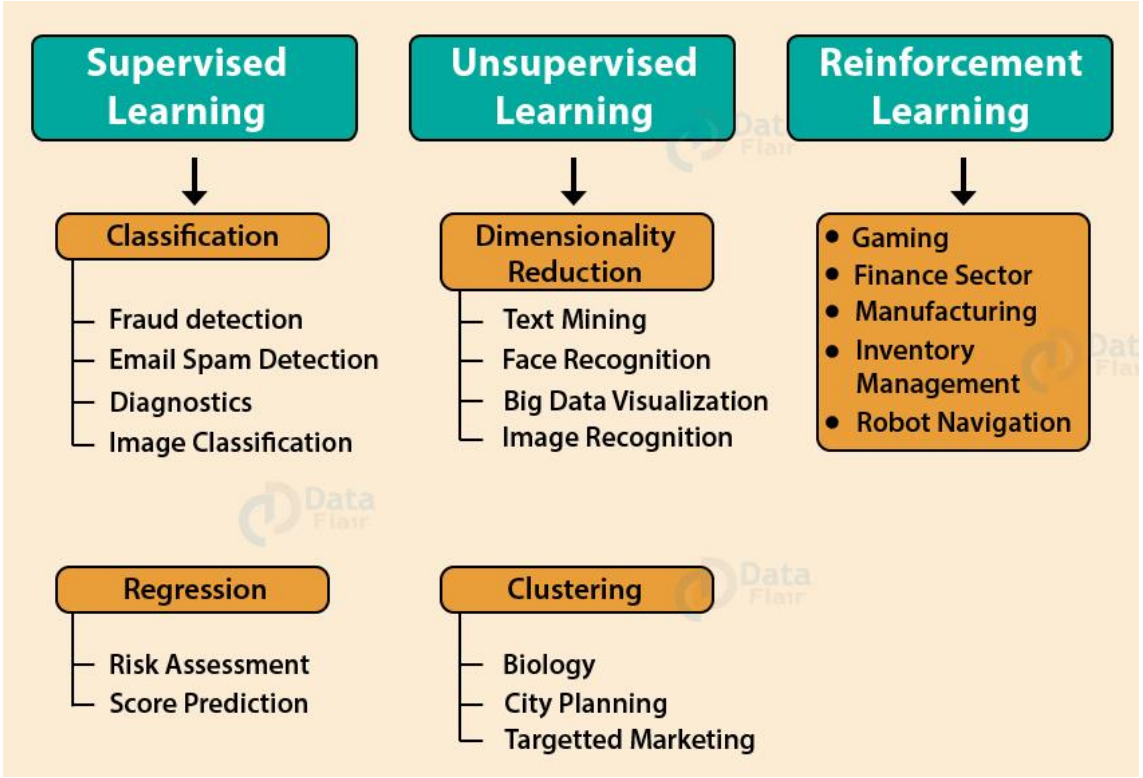
KOREATECH 한국기술교육대학교

Types of machine learning

- **Supervised**(or predictive) learning
  - learn a mapping from inputs x to outputs y
  - training set (input-output pairs) $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$

- **Unsupervised**(or descriptive) learning
  - only given inputs, $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$
  - goal : find “interesting patterns” in the data.

- **Reinforcement** learning
  - reward or punishment signals.



KOREATECH 한국기술교육대학교

지도학습

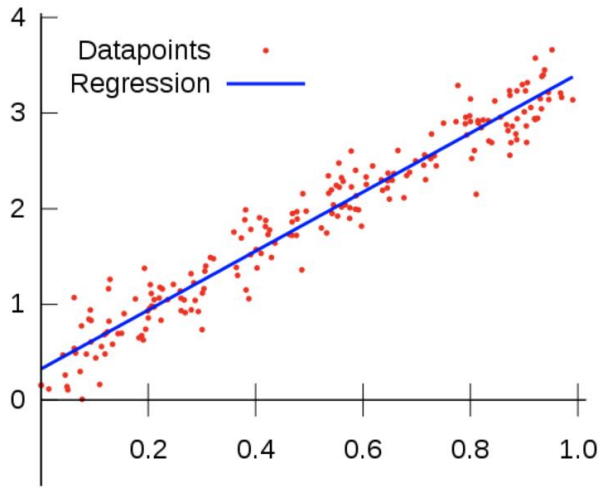
- 결과(레이블(lable))와 입력을 같이 주어서 학습하고, 주어진 입력을 분류(classification) 하거나 예측하는 회귀(regression)가 있음
- 전통적인 기계학습 알고리즘
  - 선형 회귀: Linear Regression
  - 로지스틱 회귀: Logistic Regression
  - K-최근접 이웃: K-Nearest Neighbors
  - 결정 트리: Decision Tree
  - 랜덤 포레스트: Random Forest
  - 서포트 벡터 머신: Support Vector Machine

선형회귀

- 가장 기본적인 알고리즘
- 데이터들과 오차가 가장 적은 회귀선 생성

$$y = a_1 x + a_0$$

주어진 x, y 값을 이용해서 a_0, a_1 를 구함



$$S(a_0, a_1) = \sum_{i=0}^{n} [y_i - f(x_i)]^2 = \sum_{i=0}^{n} (y_i - a_0 - a_1 x_i)^2$$

$$\frac{\partial S}{\partial a_0} = \sum_{i=0}^{n} -2(y_i - a_0 - a_1 x_i) = 2\left[a_0(n+1) + a_1 \sum_{i=0}^{n} x_i - \sum_{i=0}^{n} y_i\right] = 0$$

$$\frac{a_0(n+1)}{n+1} + a_1 \frac{\sum_{i=0}^{n} x_i}{n+1} - \frac{\sum_{i=0}^{n} y_i}{n+1} = 0$$

$$a_0 = \bar{y} - a_1 \bar{x}$$

KOREATECH 한국기술교육대학교

선형회귀

$$\frac{\partial S}{\partial a_1} = \sum_{i=0}^{n} -2(y_i - a_0 - a_1 x_i)x_i = 2\left[a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i\right] = 0$$

$$a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i = 0$$

$$(\bar{y} - a_1 \bar{x}) \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i = 0$$

$$\sum_{i=0}^{n} y_i \bar{x} - a_1 \sum_{i=0}^{n} x_i \bar{x} + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i = 0$$

$$a_1 \sum_{i=0}^{n} x_i (x_i - \bar{x}) = \sum_{i=0}^{n} y_i (x_i - \bar{x}) \qquad a_1 = \frac{\sum_{i=0}^{n} y_i (x_i - \bar{x})}{\sum_{i=0}^{n} x_i (x_i - \bar{x})}$$

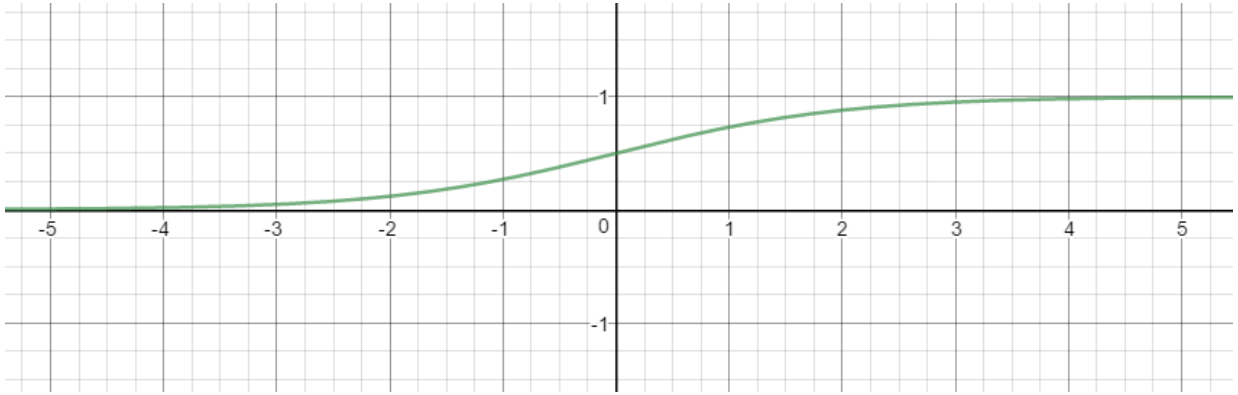
다중선형회귀 $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$

KOREATECH 한국기술교육대학교

로지스틱 회귀

❖ 출력결과를 0과 1사이로 변환

$$y = \frac{1}{1 + e^{-x}}$$



❖ 이항분류

$$P(Y = 1|X = \vec{x}) = \frac{1}{1 + e^{-\vec{\beta}^T \vec{x}}}$$

❖ 다항분류

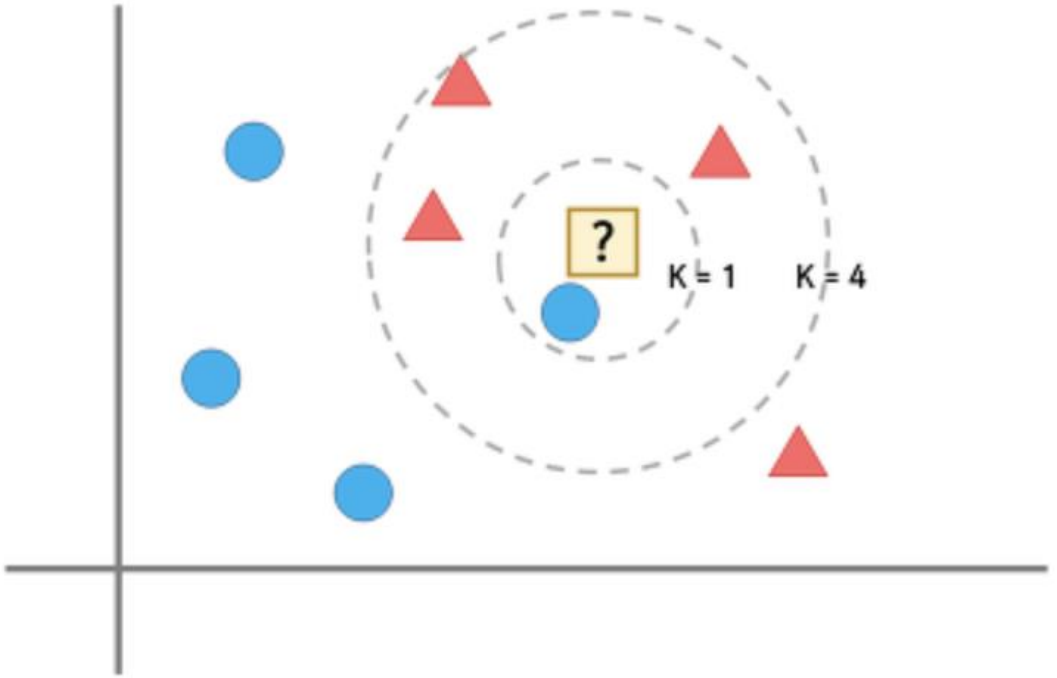
$$P(Y = k|X = \vec{x}) = \frac{e^{\vec{\beta_k}^T \vec{x}}}{1 + \sum_{i=1}^{K-1} e^{\vec{\beta_i}^T \vec{x}}} \quad (k = 0, 1, \ldots, K-1)$$

$$P(Y = K|X = \vec{x}) = \frac{1}{1 + \sum_{i=1}^{K-1} e^{\vec{\beta_i}^T \vec{x}}}$$

KOREATECH
한국기술교육대학교

K-Nearest Neighbore

- ❖ 새로운 데이터를 입력 받았을 때 어디에 속하는지 결정하는 알고리즘
- ❖ ?는 어디에 가까운가? (k 반경에 있는 데이터들의 거리 제곱근의합)

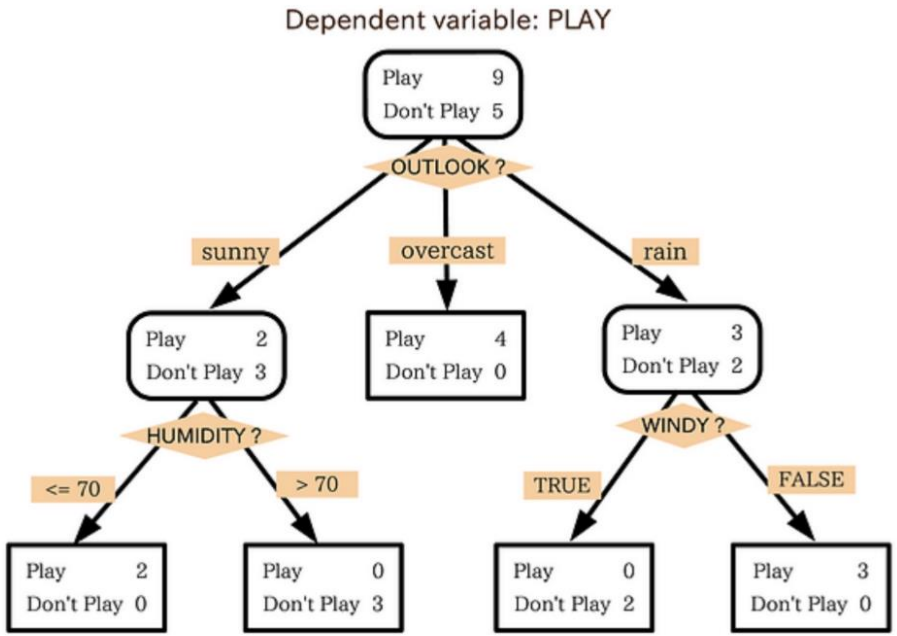


KOREATECH
한국기술교육대학교

결정 트리(Decision Tree)

❖ 운동경기가 진행여부 판단

- 비가오지만 바람이 불지 않으면 경기가 열림
- 맑은날이지만 습도가 높으면 경기가 열리지 않음

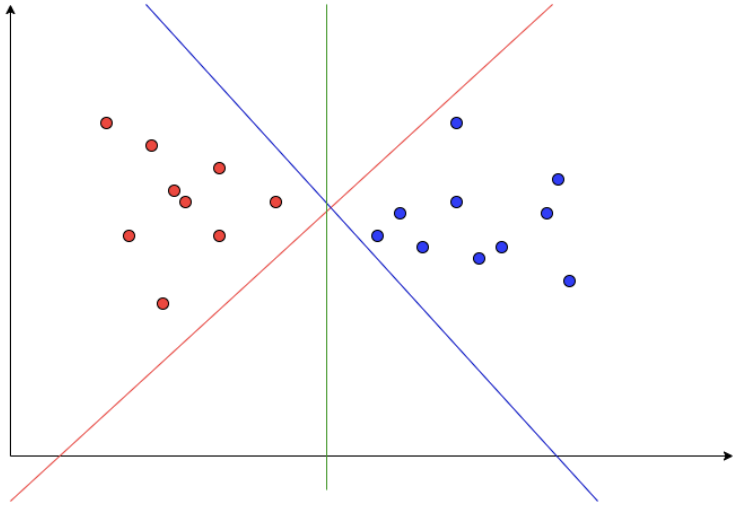


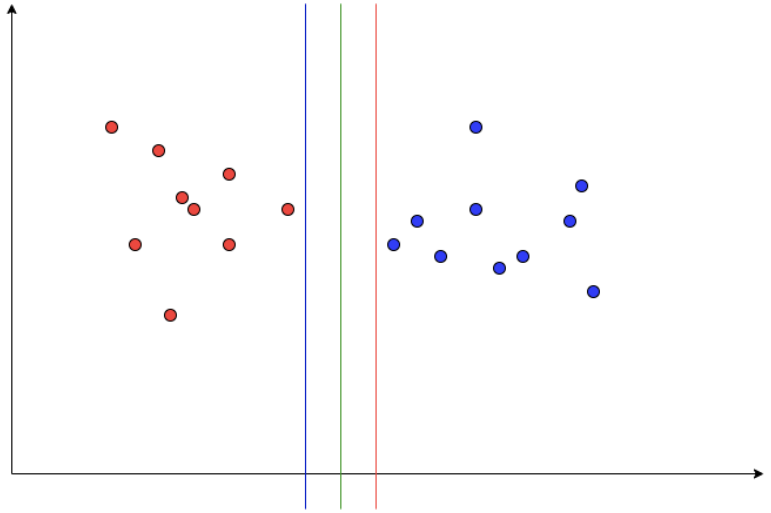
❖ 랜덤 포레스트는 결정트리들의 모임

KOREATECH 한국기술교육대학교

SVM

❖ n 차원을 n-1 차원으로 나눌 수 있다.



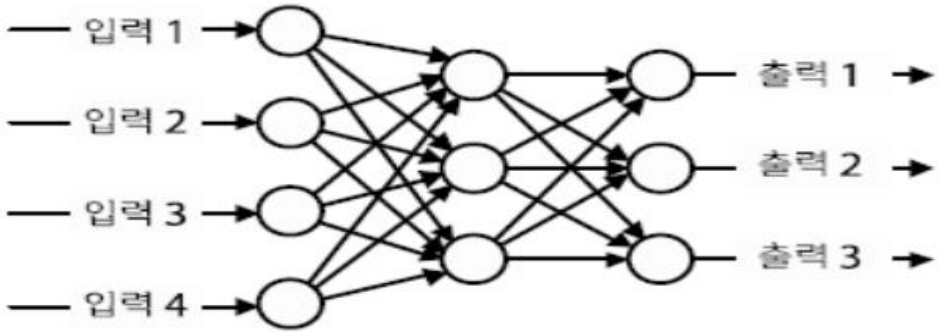


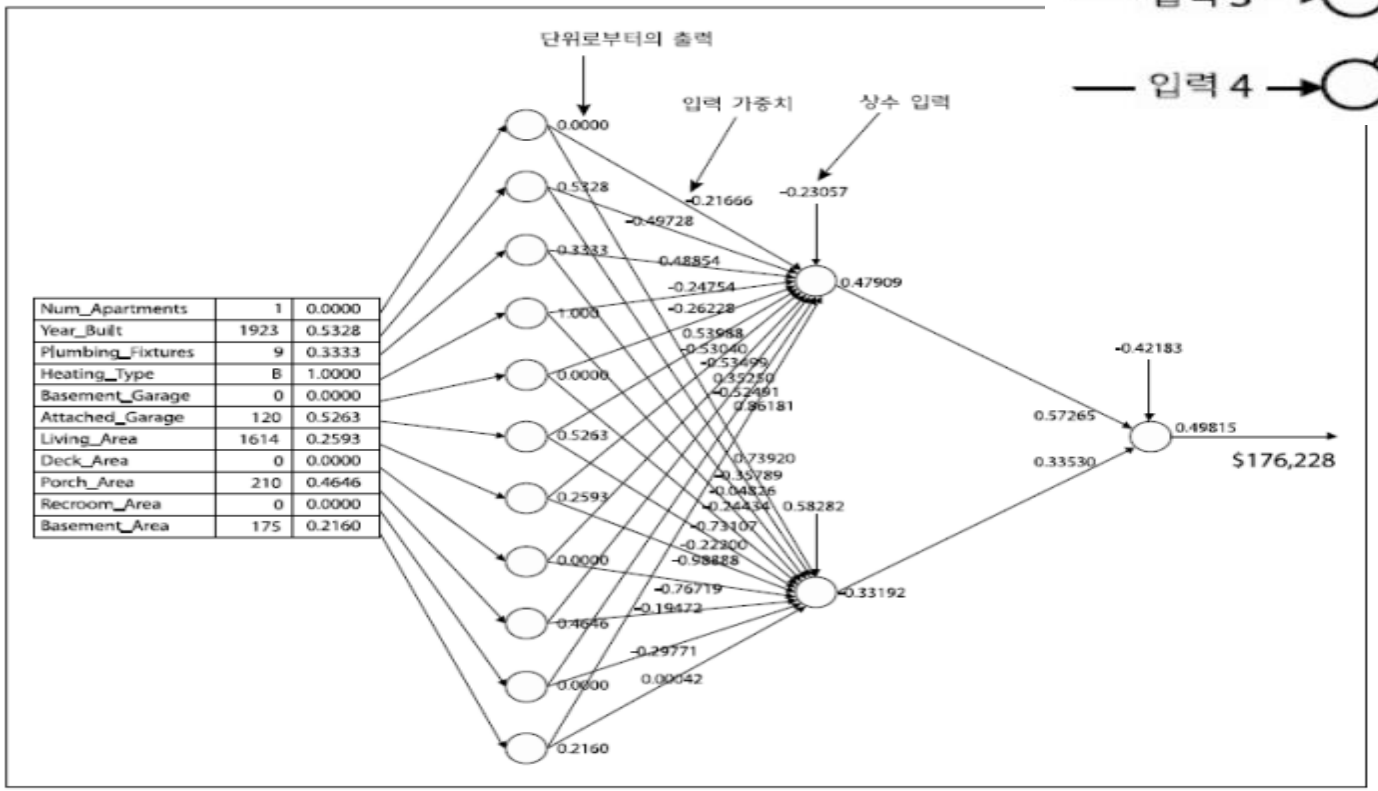
❖ 저차원에서 선형분리가 안되는 것은 고차원을 확장해서 선형분리를 수행하고 저차원으로 환원

KOREATECH 한국기술교육대학교

인공신경망

❖ 다층네트워크





출처: 김종우, 김선태, 경영을 위한 데이터마이닝, 한경사, 2009

KOREATECH 한국기술교육대학교

학습정리

- ❖ Markdown 응용(Note-taking APP)
  - Obsidian (https://obsidian.md/)
  - Logseq (https://logseq.com/)
- ❖ 인공지능/미래직업
- ❖ 기계학습