

# ML\_1

Team BMS



# INDEX

ML Study  
1 Week

---

## Index 01. Ice Breaking

Ice Breaking

---

## Index 02. Chapter 1

한 눈에 보는 머신러닝

---

## Index 03. Chapter 2

머신러닝 프로젝트 처음부터 끝까지

---



# Ice Breaking

# Ice Breaking

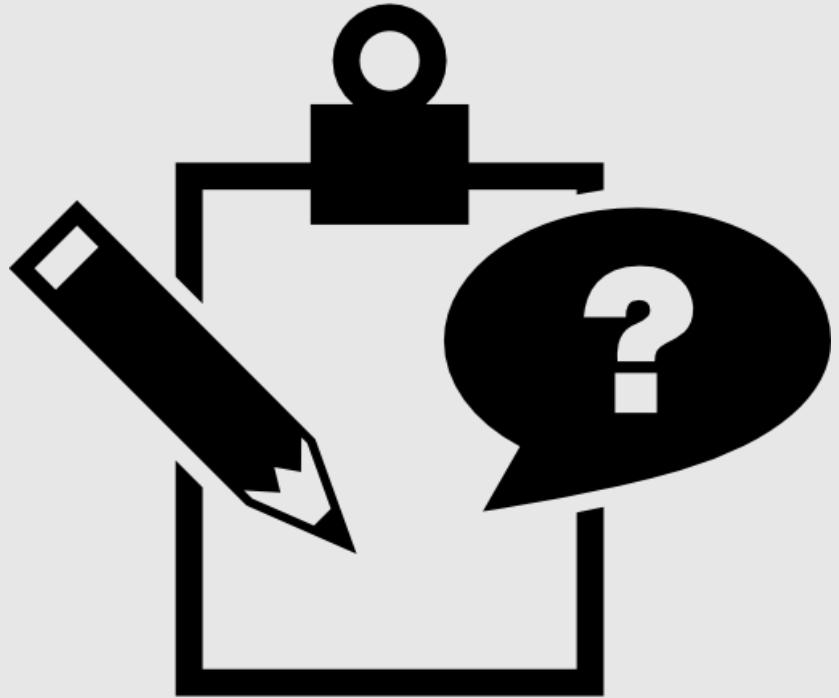


noahcha

약 1분 전

어느 아내가 프로그래머 남편에게 「쇼핑하러 갈 때, 우유 하나 사와, 아, 계란 있으면 6개 사와」 남편은 잠시 후, 우유를 6개 사왔다. 아내는 물었다, 「왜 우유를 6개나 사왔어!」 남편 「계란이 있길래 6개 사왔지…」

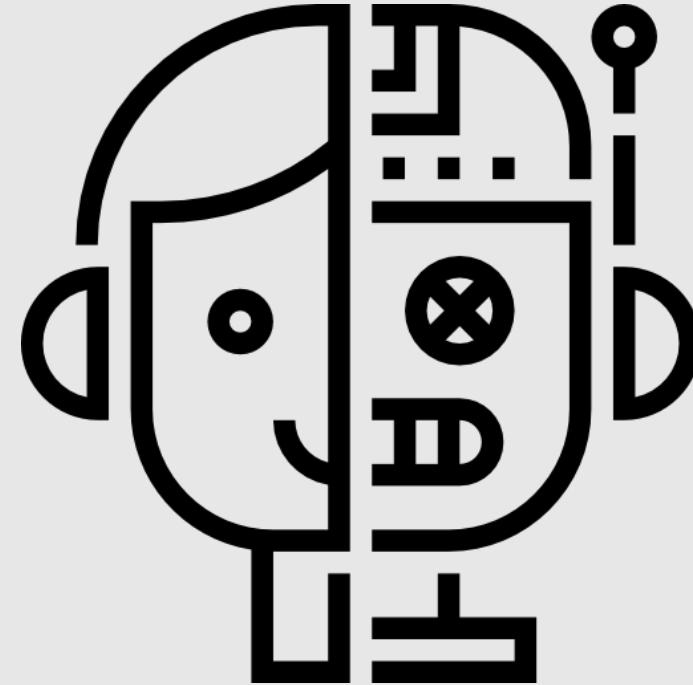
twtkr에서 작성된 글



# The Machine Learning Landscape

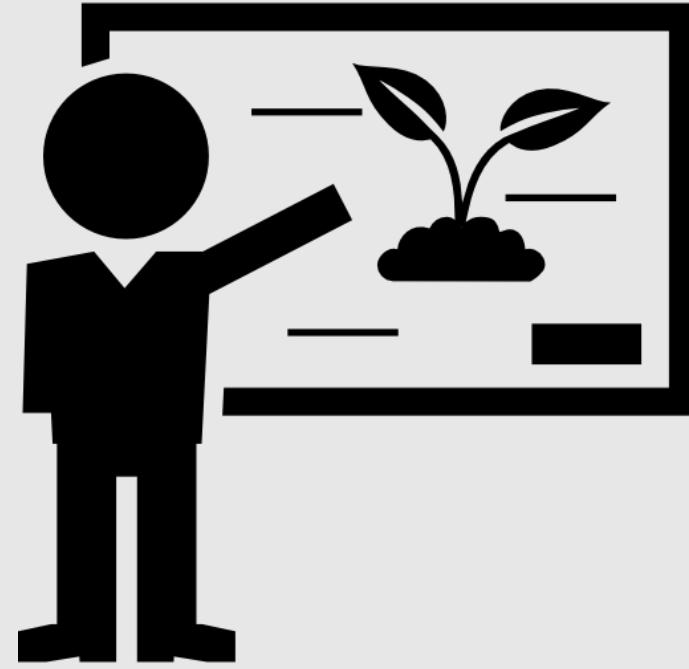
01 I. The Machine Learning Landscape

# Artificial Intelligence



기계로부터 만들어진 지능

# Machine Learning



기계를 가르침 =  
Machine Learning

# Machine Learning

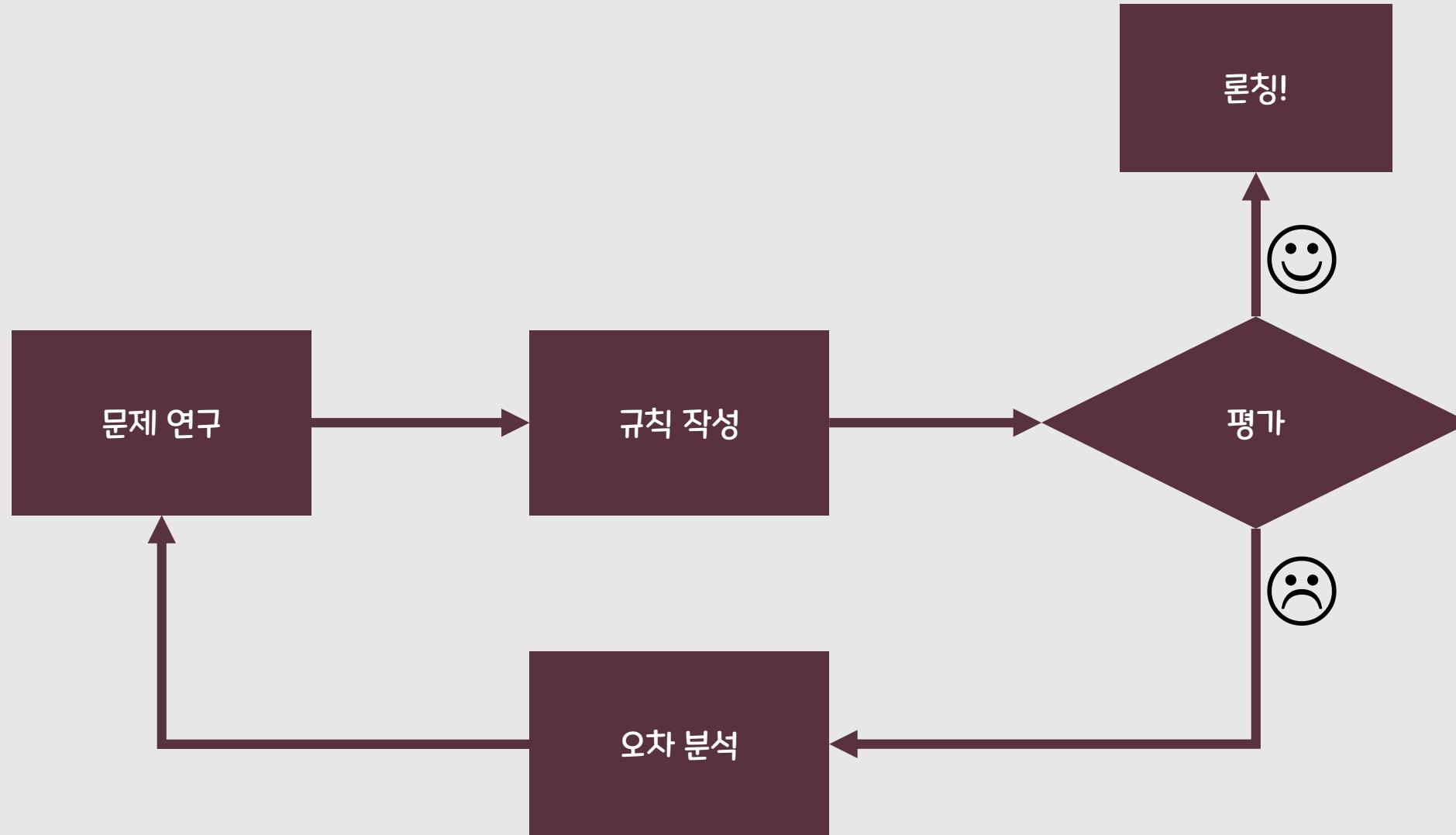
A computer program is said to learn if its performance at a task  $T$ , as measured by a performance  $P$ , improves with experience  $E$ .

- Tom Mitchell, 1997

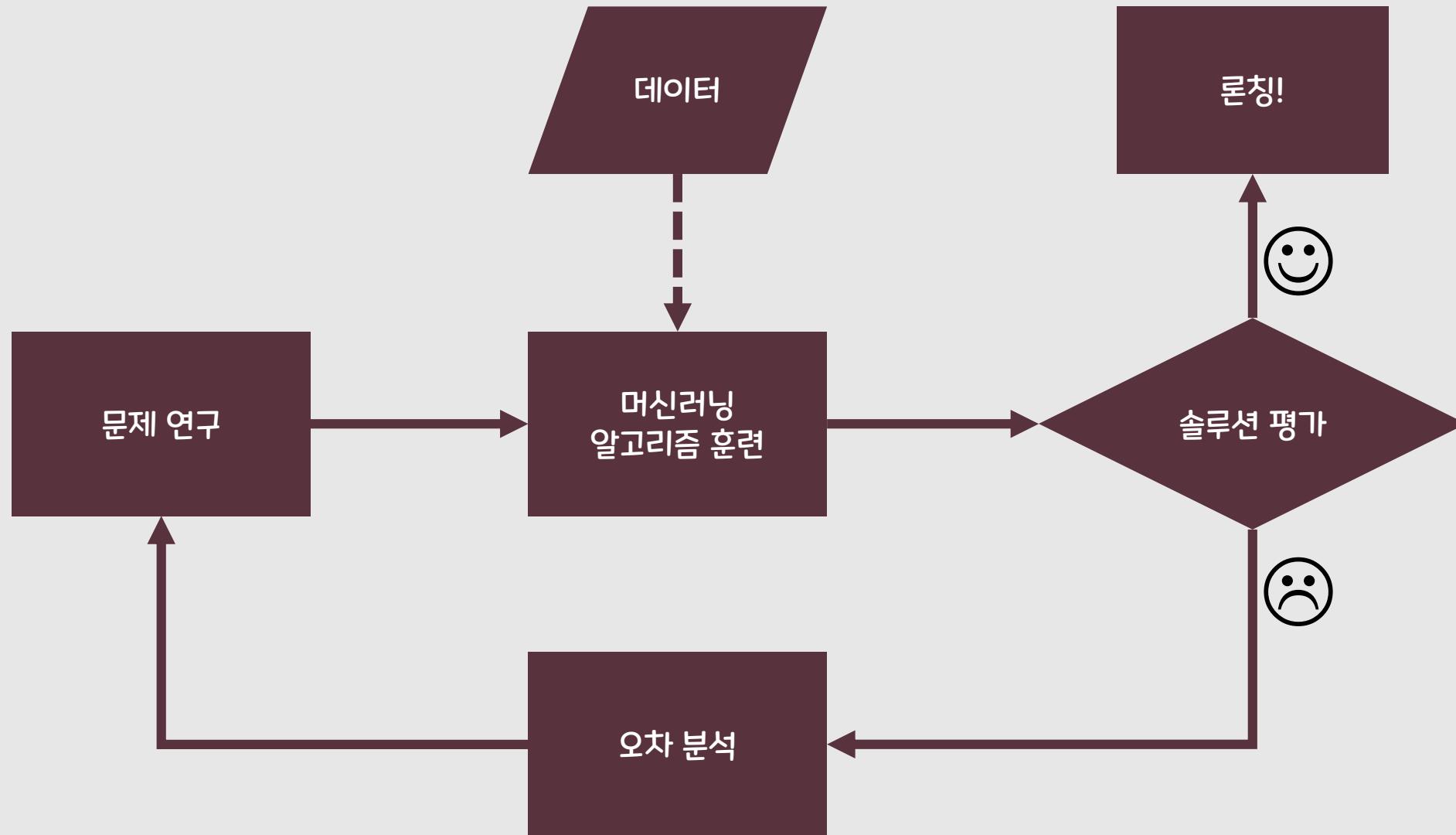
(어떤 작업  $T$ 에 대한 컴퓨터 프로그램의 성능을  $P$ 로 측정했을 때 경험  $E$ 로 인해 성능이 향상됐다면, 이 컴퓨터 프로그램은 작업  $T$ 와 성능 측정  $P$ 에 대해 경험  $E$ 로 학습한 것이다.)

01 I. The Machine Learning Landscape

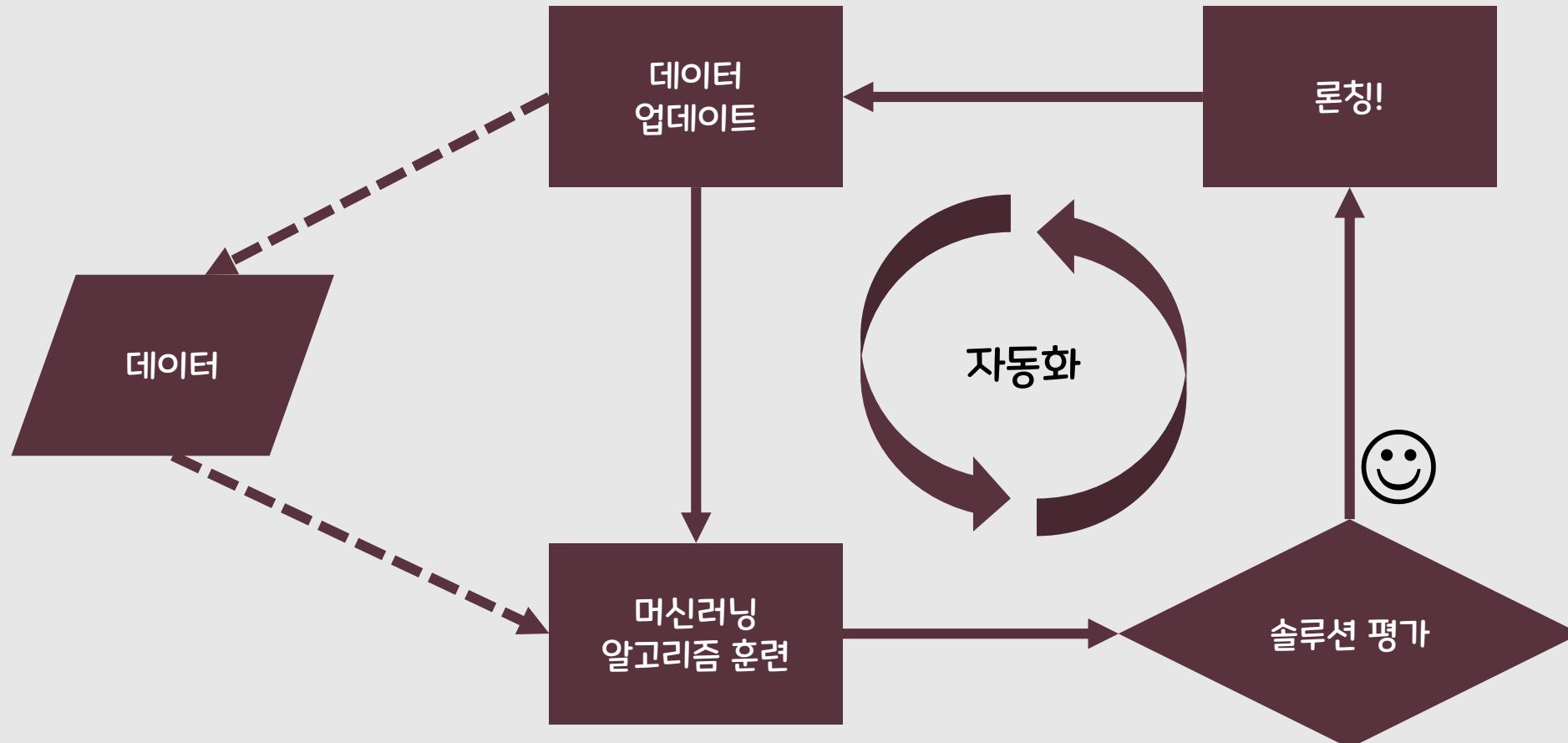
# Why Machine Learning?



# Why Machine Learning?

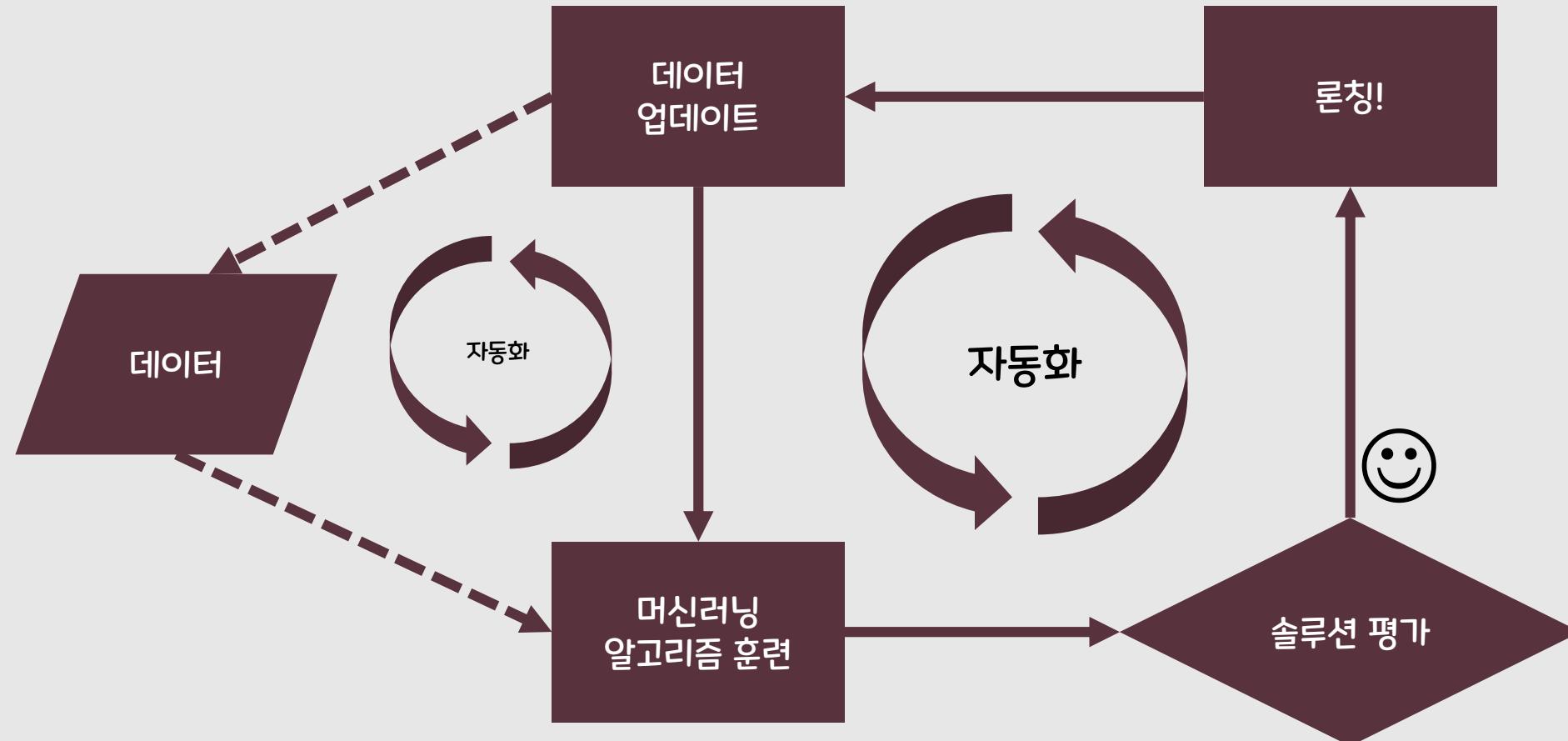


# Why Machine Learning?

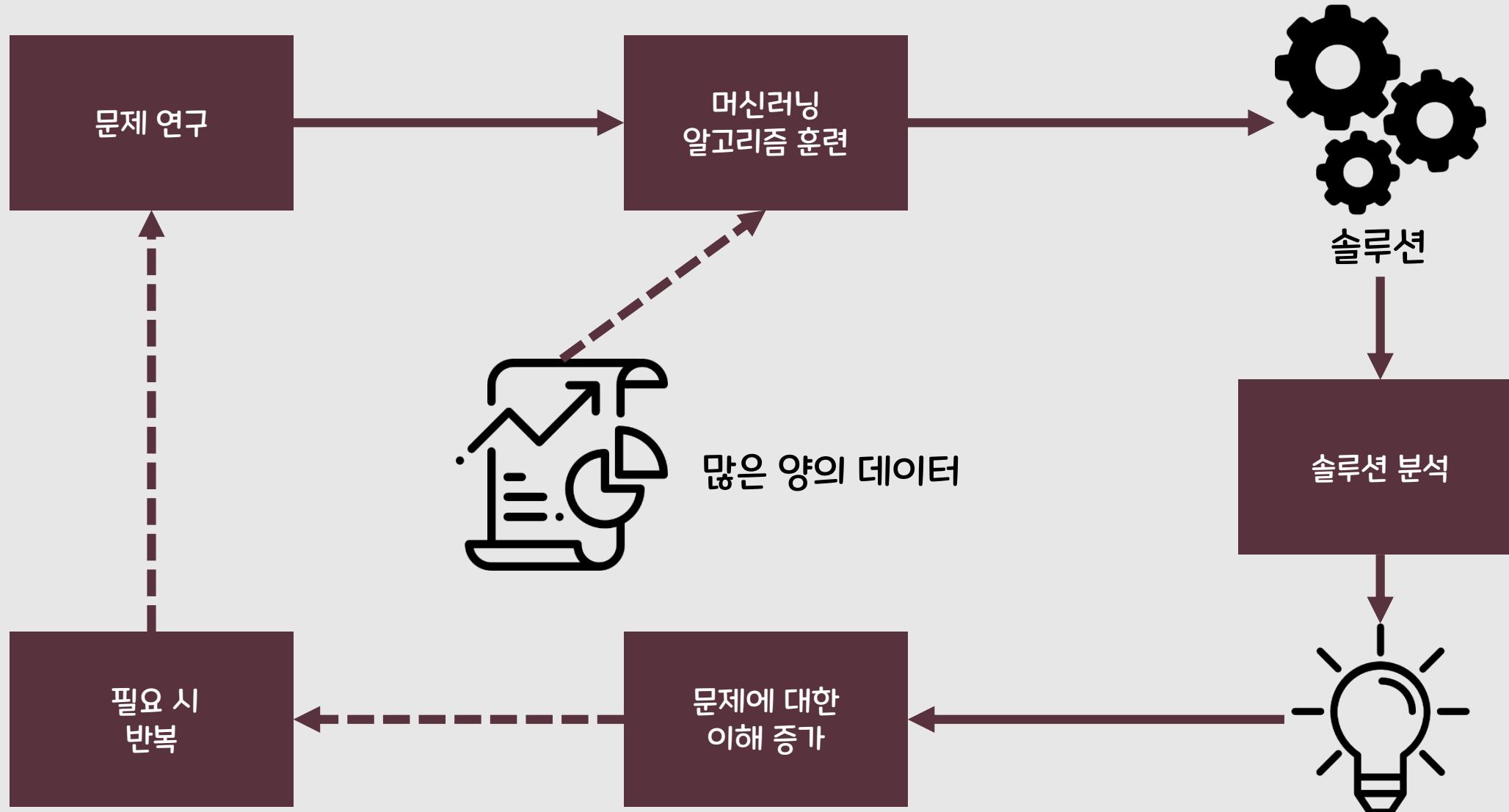


01 I. The Machine Learning Landscape

# Auto ML



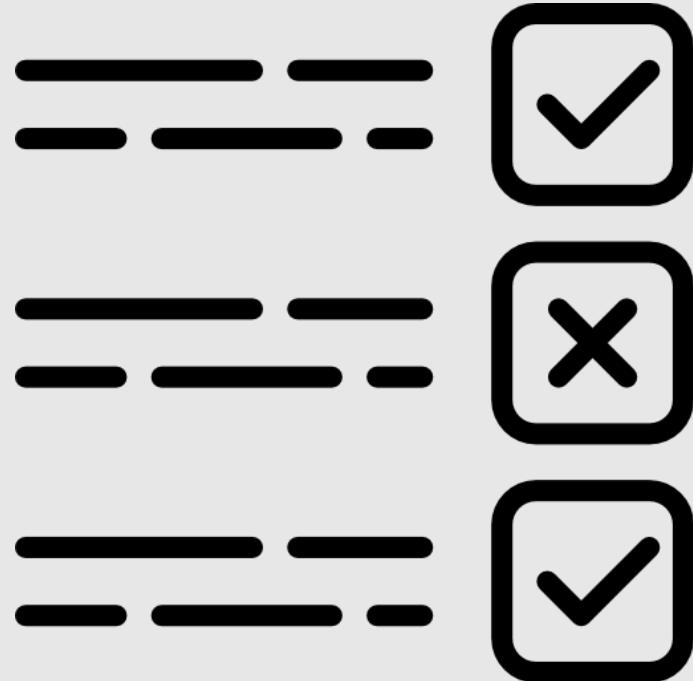
# Data Mining



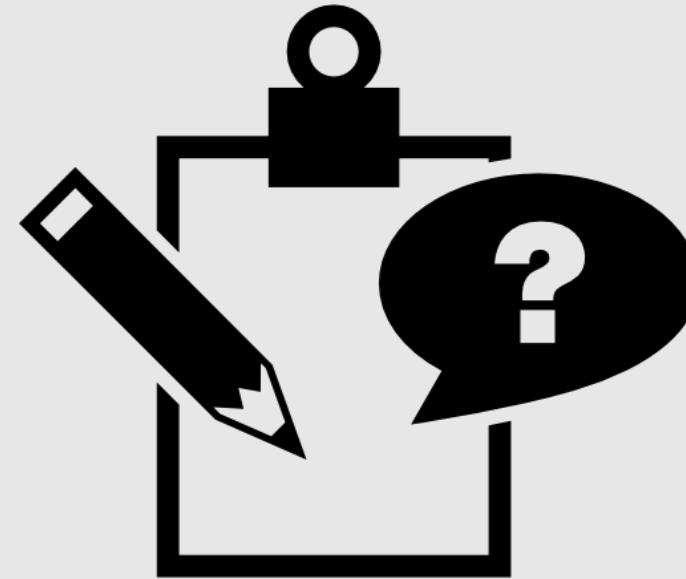
## Kinds of Machine Learning

- 사람의 감독 하에 훈련하는 것인지 그렇지 않은 것인지
  - 실시간으로 점진적인 학습을 하는지 아닌지
- 단순하게 알고 있는 데이터 포인트와 새 데이터 포인트를 비교하는 것인지 아니면 훈련 데이터셋에서 과학자들처럼 패턴을 발견하여 예측 모델을 만드는 것인지

## Kinds of Machine Learning I



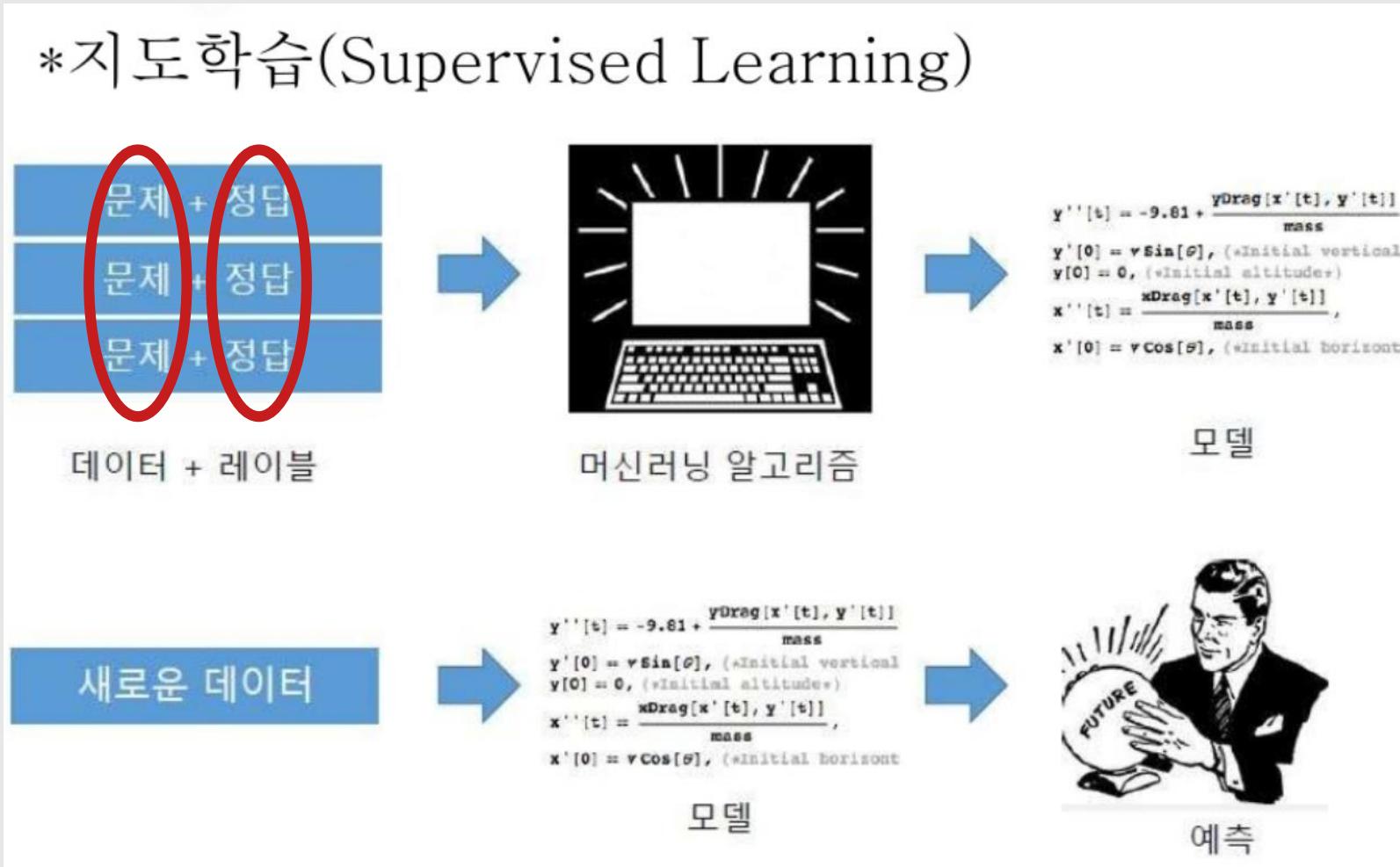
지도 학습



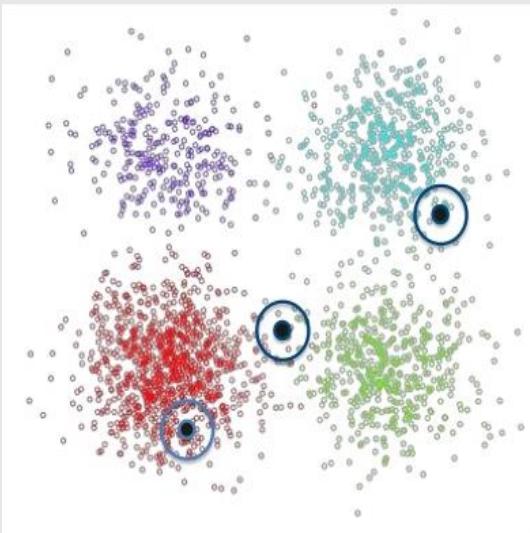
비지도 학습

01 I. The Machine Learning Landscape

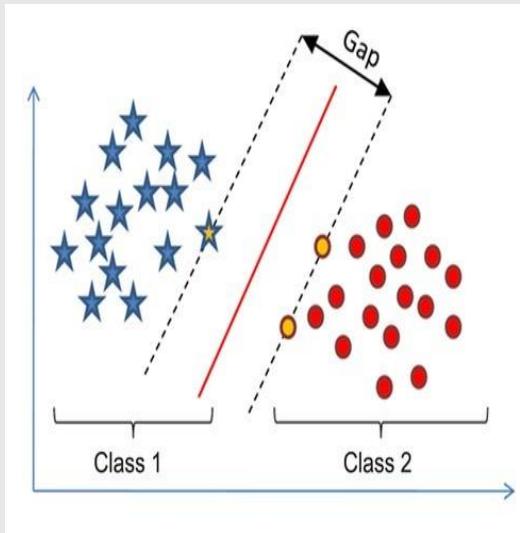
# Supervised Learning



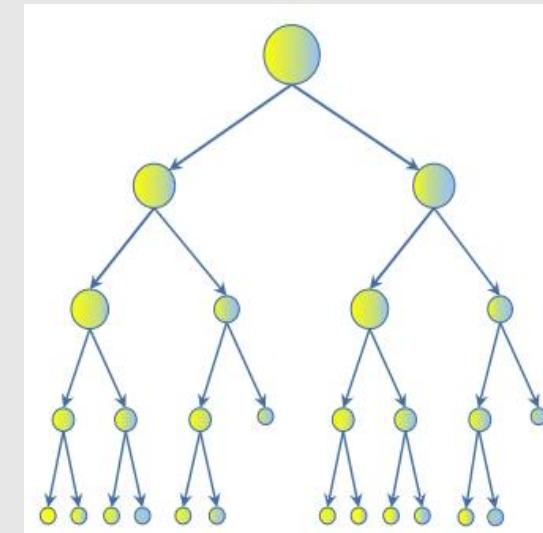
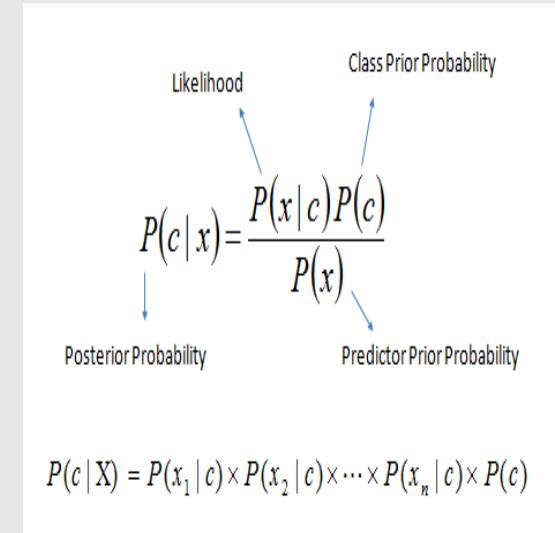
# Supervised Learning



kNN



SVM

Decision  
Tree

# Supervised Learning



분류

(예: 손 글씨 분류)



회귀

(예: 집값 예측)

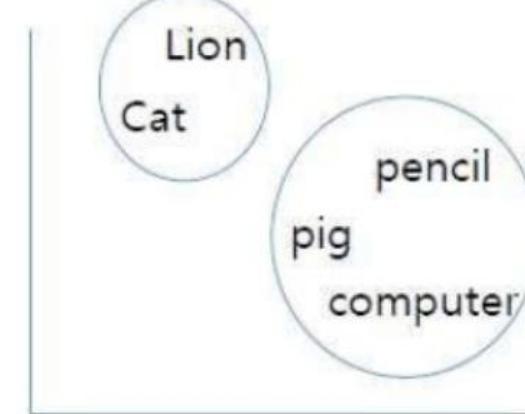
## Unsupervised Learning

\*자율학습 (Unsupervised Learning)  
=비지도학습

|          |
|----------|
| Cat      |
| computer |
| Lion     |
| pencil   |
| pig      |



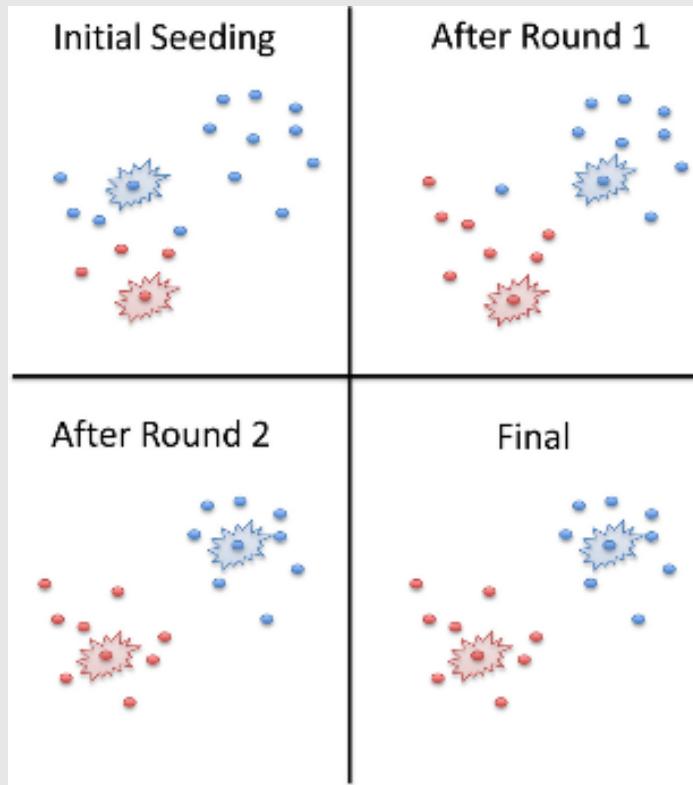
머신러닝 알고리즘



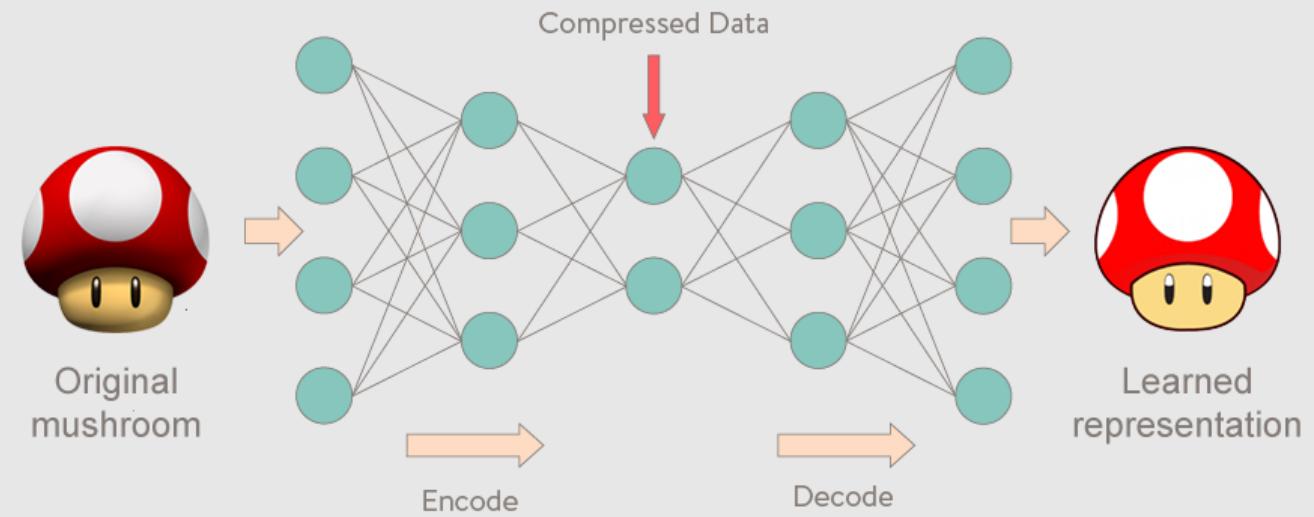
그룹

알려지지 않은 데이터

# Unsupervised Learning



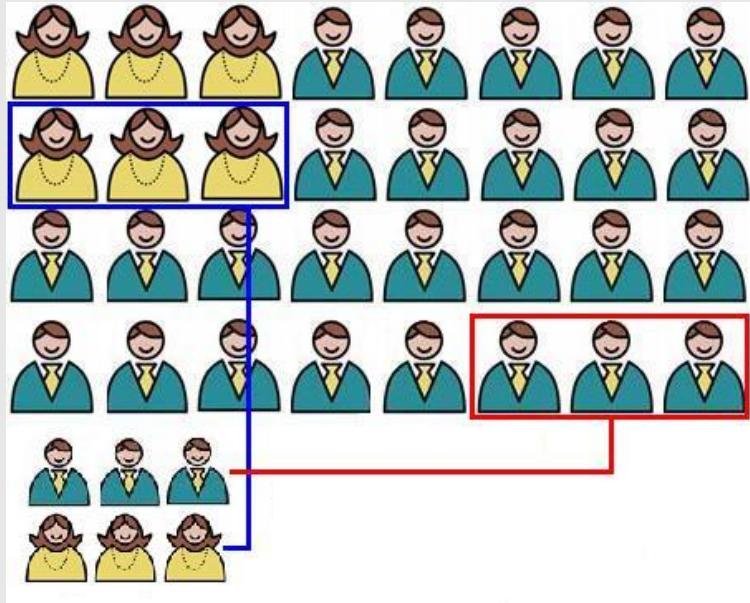
kMeans



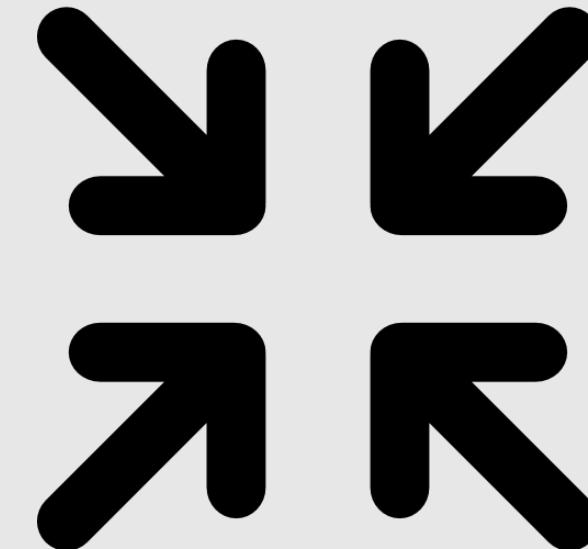
Autoencoder

01 I. The Machine Learning Landscape

# Unsupervised Learning



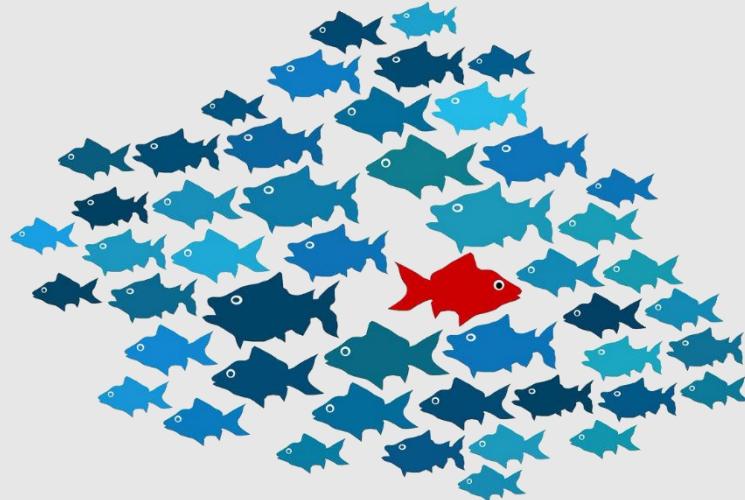
군집화  
(예: 남녀 구  
분)



차원 축소 및 시각화

01 I. The Machine Learning Landscape

# Unsupervised Learning

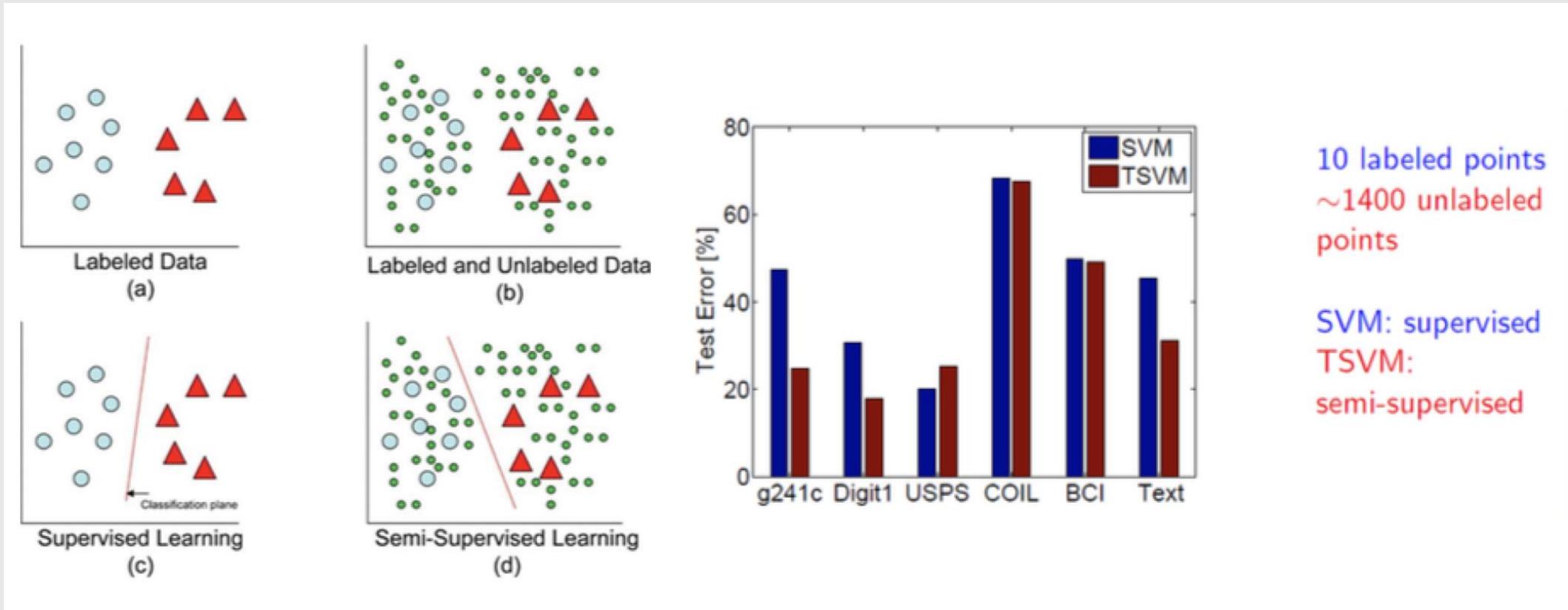


이상치 탐지

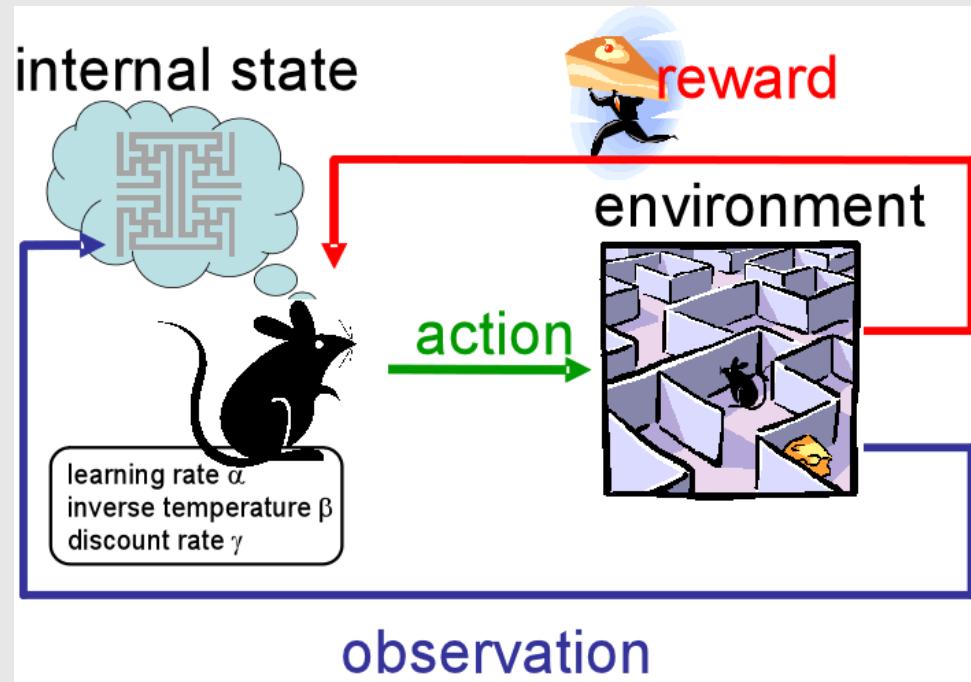


연관규칙 생성

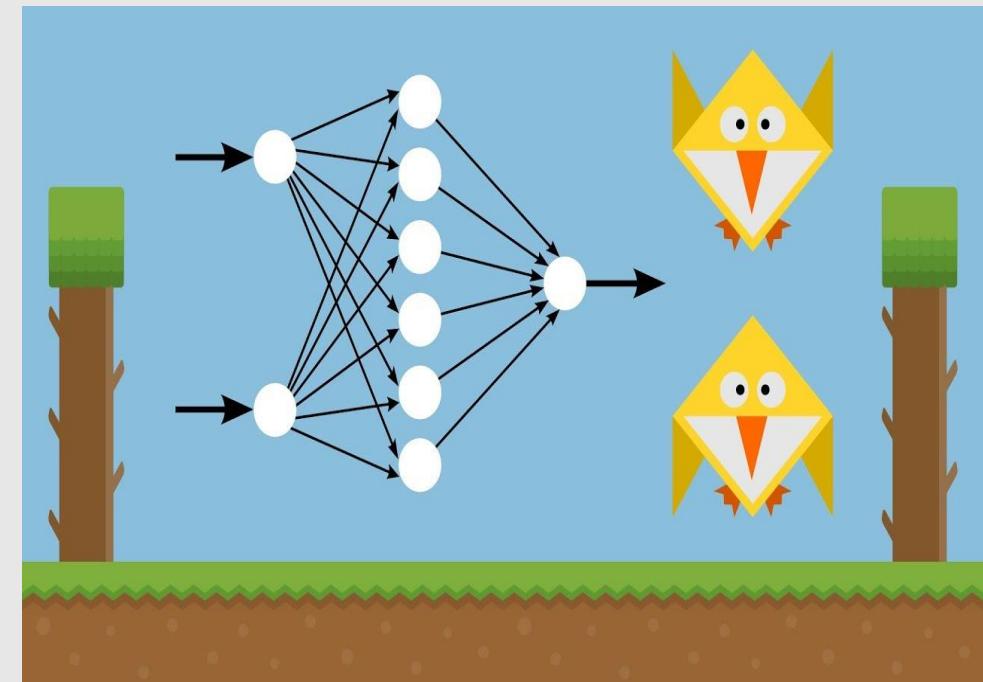
# Semi-supervised Learning



## Other Method



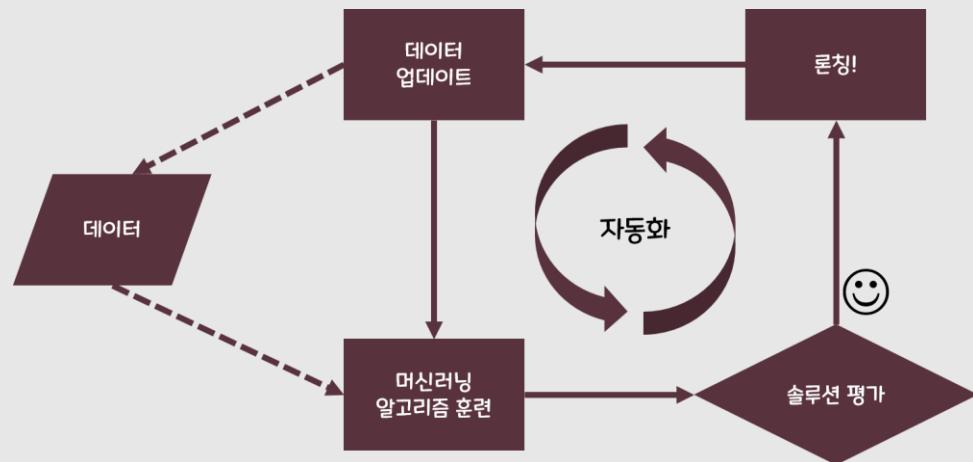
Reinforcement Learning  
(강화학습)



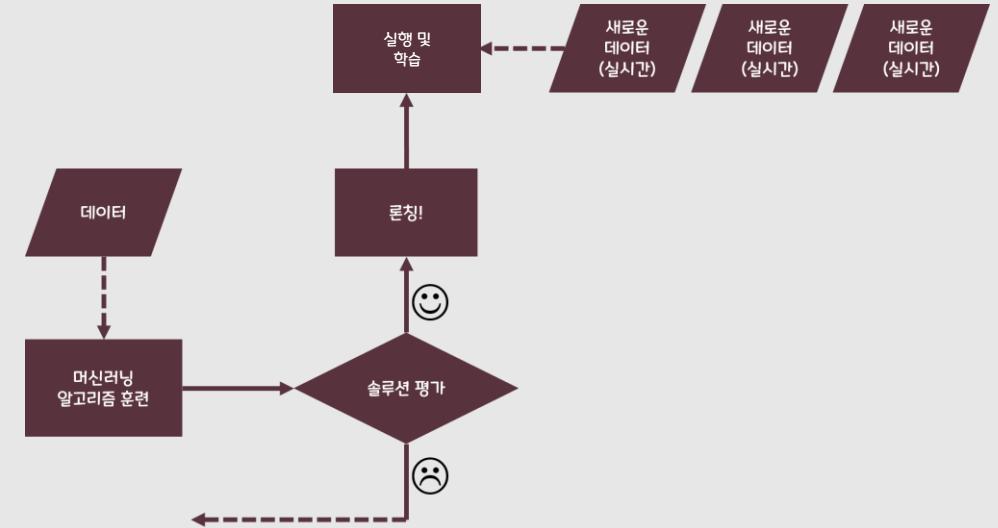
Genetic Learning  
(진화학습)

01 I. The Machine Learning Landscape

# Batch Learning & Online Learning



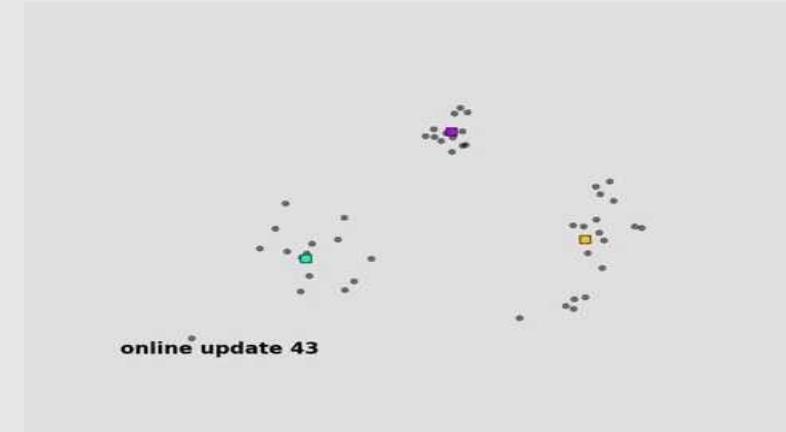
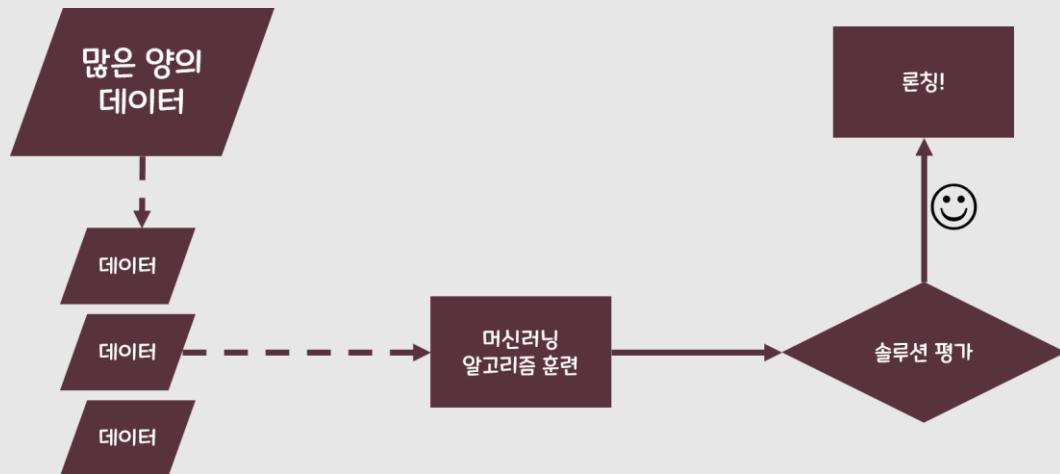
Batch Learning  
(Offline Learning)



Mini-Batch Learning  
(Online Learning)

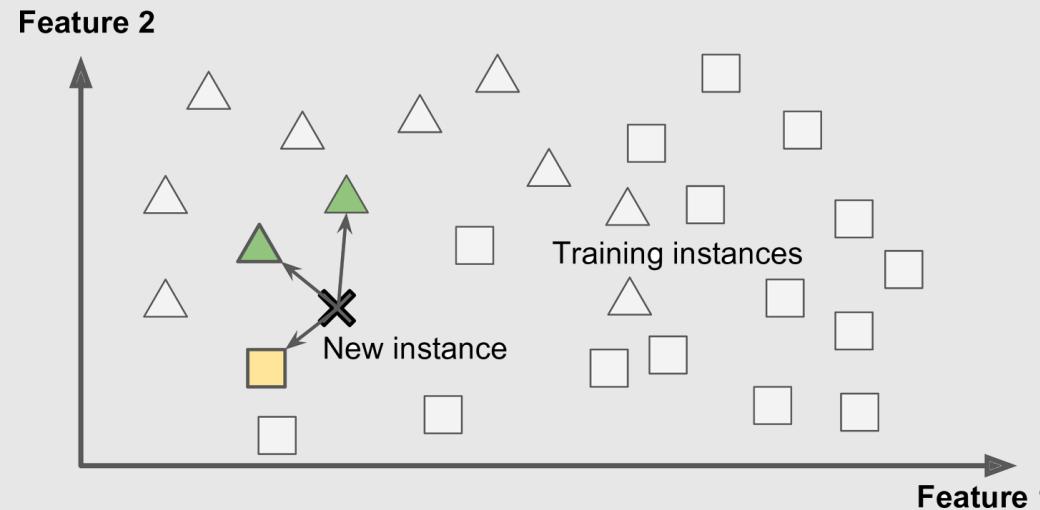
01 I. The Machine Learning Landscape

# Online Learning

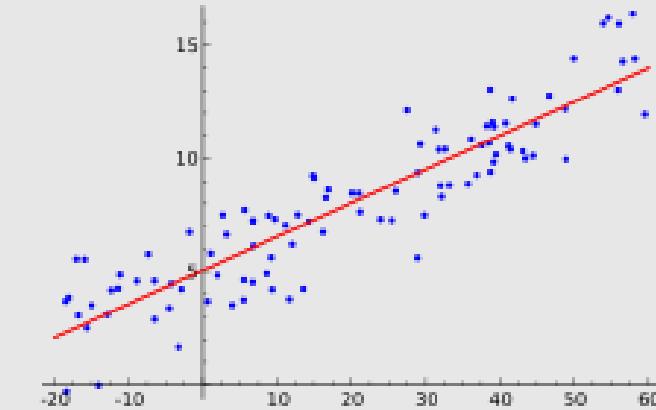


<https://www.youtube.com/watch?v=hzGnnx0k6es>

# Instance-based Learning & Model-based Learning



Instance-based Learning  
(사례기반 학습)



Model-based Learning  
(모델기반 학습)

## ML's Challenge

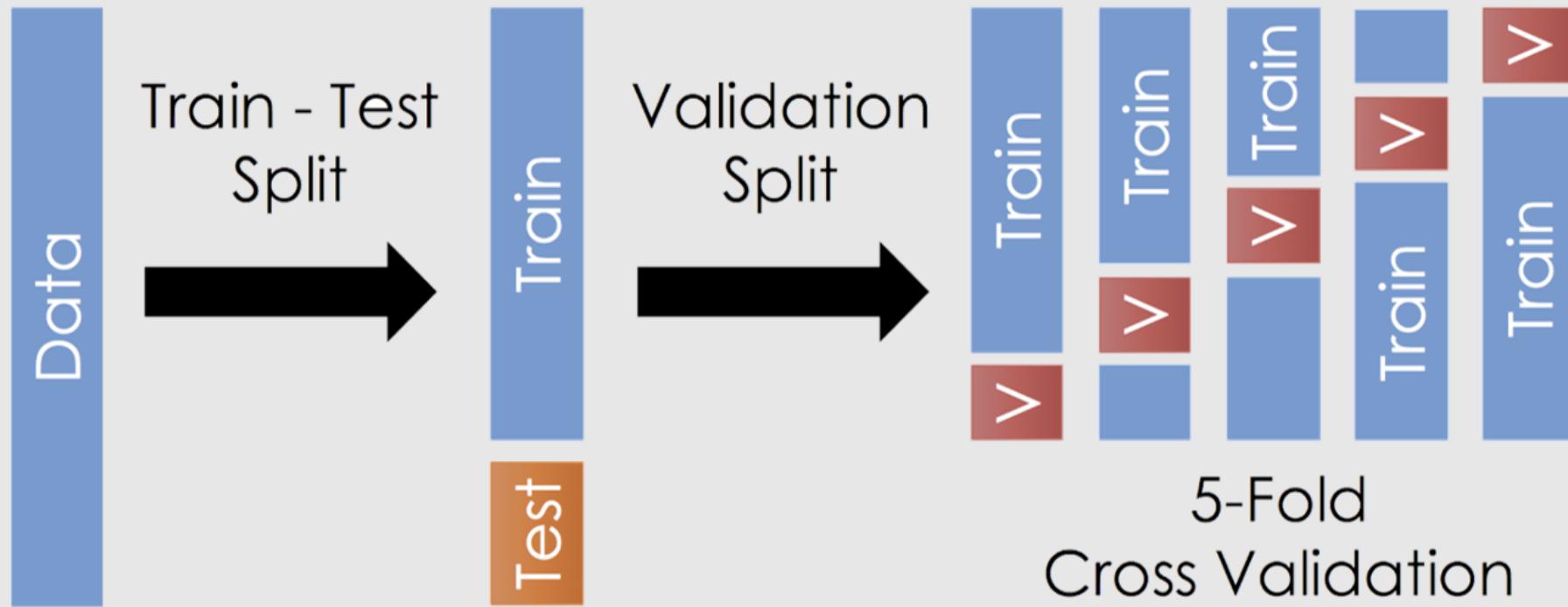
### ‘나쁜 데이터’

- 충분하지 않은 양의 데이터
  - 대표성 없는 데이터  
[샘플링 편향 및 비응답 편향]
  - 낮은 품질의 데이터
    - 관련 없는 특성  
[특성 공학(Feature Engineering)]

### ‘나쁜 알고리즘’

- 과적합 (과대적합)  
[더 많은 데이터 혹은 잡음 제거]
  - 과소적합  
[더 많은 데이터 혹은 하이퍼 파라미터 조정]

# Train & Test Split



## No Free Lunch Theorem (NFL)

데이터에 관해 완벽하게 어떤 가정도 하지 않으면  
다른 모델보다 선호할 근거가 없다.

→ 경험하기 전에 더 잘 맞을 거라고 보장할 수 있는 모델은 없다.

# No Free Lunch Theorem (NFL)

## 2.1 컴퓨터 구조에서 “The No Free Lunch Theorem”

- cpu의 발전은 과거에는 단일 프로세서의 향상을 통해 이루어졌습니다. 하지만 시간이 흐르면서 한계점이 있었고 이에 따라 단일 프로세서의 향상은 거의 없지만 프로세서의 갯수를 늘리면서 발전이 되었습니다. 이에 따라 프로그래머들은 더 이상 가만히 있어도 어플리케이션의 성능이 올라가지 않게 되었습니다. 과거에는 어플리케이션이 단일프로세서가 증가함에 따라 성능이 공짜로 증가했지만(Free lunch) 이제는 노력하여 병렬화된 프로그램을 작성하여 성능을 높여야 하게 되었죠

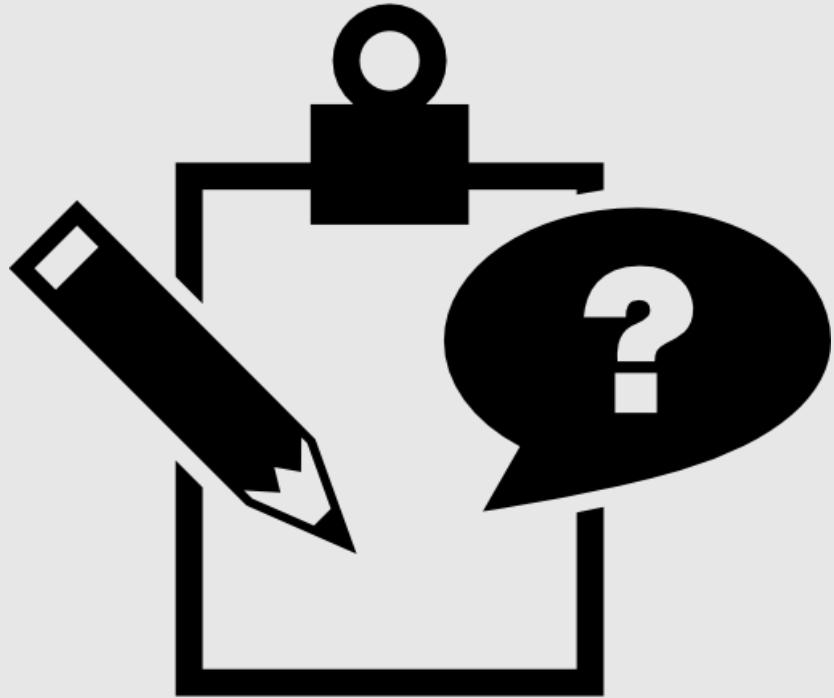
## 2.2 optimization에서 “The No Free Lunch Theorem”

- optimization이론에서는 완벽한 optimization이론은 세상에 존재할 수 없으니 함수의 특성에 맞추어서 다양한 optimization 알고리즘을 만들어야 합니다. 그 어떤 이론도 특정 문제에 대해서만 좋은 optimization알고리즘일 뿐이며 다른 문제에서는 뛰어 날 수 없다고 합니다.

“The No Free Lunch Theorem”이 머신러닝에서 의미하는 바는 무엇일까요?? general하고 universal한 AI는 불가능하니 포기해! 일까요?

아닙니다. 현실세계, 자연에서는 자연에 특화된 어떤 task가 있고 그 특성을 잘 파악해서 머신러닝 알고리즘을 구현해야 한다는 것을 의미합니다.

즉 ML연구자의 목표는 real world의 data-distribution에 대해 이해하려고 노력해야 할 것이며 이러한 노력을 바탕으로 “우리의 task(자연과 인간 사회)”에서 잘 작동하는 알고리즘을 만들내야 할 것입니다.



# End-to-End Machine Learning Project

# Introduction

## 1. 큰 그림을 봅니다.

(Look at the Big Picture)

## 2. 데이터를 구합니다.

(Get the Data)

## 3. 데이터로부터 통찰을 얻기 위해 탐색하고 시각화합니다.

(Discover and Visualize the Data to Gain Insights)

## 4. 머신러닝 알고리즘을 위해 데이터를 준비합니다.

(Prepare the Data for Machine Learning Algorithms)

## 5. 모델을 선택하고 훈련시킵니다.

(Select and Train a Model)

## 6. 모델을 상세하게 조정합니다.

(Fine-Tune Your Model)

## 7. 솔루션을 제시합니다.

(Present a Solution)

## 8. 시스템을 론칭하고 모니터링하고 유지 보수합니다.

(Launch, Monitor, and Maintain Your System)

## Look at the Big Picture

1. 목표를 비즈니스 용어로 정의합니다.
2. 이 솔루션은 어떻게 사용될 것인가?
3. (만약 있다면) 현재 솔루션이나 차선책은 무엇인가?
4. 어떤 문제라고 정의할 수 있나 (지도/비지도, 온라인/오프라인 등)?
  5. 성능을 어떻게 측정해야 하나?
6. 성능 지표가 비즈니스 목표에 연결되어 있나?
7. 비즈니스 목표에 도달하기 위해 필요한 최소한의 성능은 얼마인가?
8. 비슷한 문제가 있나? 이전의 방식이나 도구를 재사용 할 수 있나?
  9. 해당 분야의 전문가가 있나?
10. 수동으로 문제를 해결하는 방법은 무엇인가?
11. 여러분이 세운 가정을 나열합니다.
12. 가능하면 가정을 검정합니다.

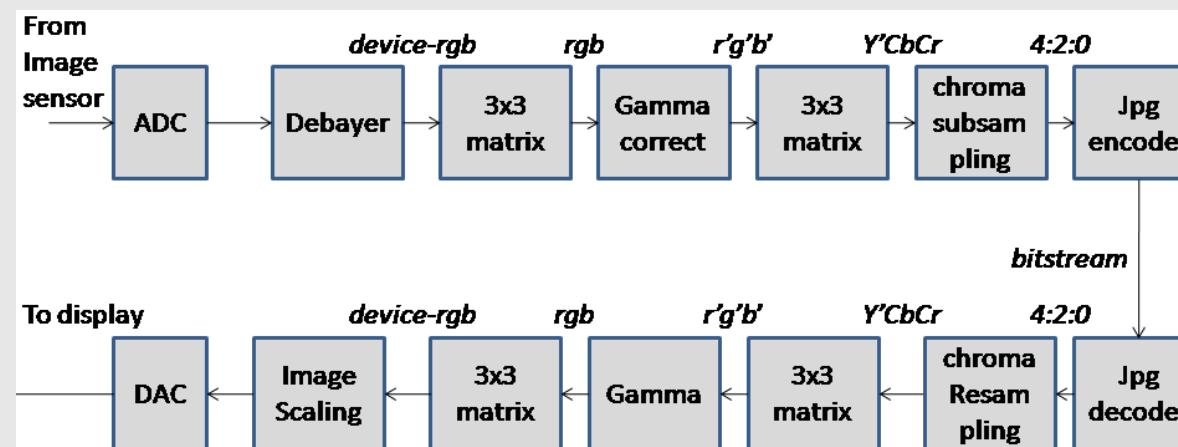
## Look at the Big Picture

1. 목표를 비즈니스 용어로 정의합니다.
2. 이 솔루션은 어떻게 사용될 것인가?
3. (만약 있다면) 현재 솔루션이나 차선책은 무엇인가?
4. 어떤 문제라고 정의할 수 있나 (지도/비지도, 온라인/오프라인 등)?
  5. 성능을 어떻게 측정해야 하나?
  6. 성능 지표가 비즈니스 목표에 연결되어 있나?
7. 비즈니스 목표에 도달하기 위해 필요한 최소한의 성능은 얼마인가?
8. 비슷한 문제가 있나? 이전의 방식이나 도구를 재사용 할 수 있나?
  9. 해당 분야의 전문가가 있나?
10. 수동으로 문제를 해결하는 방법은 무엇인가?
11. 여러분이 세운 가정을 나열합니다.
12. 가능하면 가정을 검정합니다.

# Pipeline

컴퓨터 과학에서 **파이프라인**(영어: pipeline)은 한 데이터 처리 단계의 출력이 다음 단계의 입력으로 이어지는 형태로 연결된 구조를 가리킨다. 이렇게 연결된 데이터 처리 단계는 한 여러 단계가 서로 동시에, 또는 병렬적으로 수행될 수 있어 효율성의 향상을 꾀할 수 있다. 각 단계 사이의 입출력을 중계하기 위해 **버퍼**가 사용될 수 있다.

데이터 처리 컴포넌트들이 연속되어 있는 것을  
데이터 파이프라인이라고 한다.



## Look at the Big Picture

1. 목표를 비즈니스 용어로 정의합니다.
2. 이 솔루션은 어떻게 사용될 것인가?
3. (만약 있다면) 현재 솔루션이나 차선책은 무엇인가?
4. 어떤 문제라고 정의할 수 있나 (지도/비지도, 온라인/오프라인 등)?
  5. 성능을 어떻게 측정해야 하나?
  6. 성능 지표가 비즈니스 목표에 연결되어 있나?
7. 비즈니스 목표에 도달하기 위해 필요한 최소한의 성능은 얼마인가?
8. 비슷한 문제가 있나? 이전의 방식이나 도구를 재사용 할 수 있나?
  9. 해당 분야의 전문가가 있나?
10. 수동으로 문제를 해결하는 방법은 무엇인가?
11. 여러분이 세운 가정을 나열합니다.
12. 가능하면 가정을 검정합니다.

## Performance Index

### 1. MAE (Mean Absolute Error)

→ 평균 절대 오차, L1 Norm, 특이값에 덜 민감함

### 2. RMSE (Root Mean Squared Error)

→ MSE의 제곱근, 가장 보편적으로 사용, L2 Norm

### 3. RAE (Relative Absolute Error)

→ 상대적 절대 오차, 절대값을 쓰기 때문에 마찬가지로 특이값에 덜 민감함

### 4. Relative Squared Error (RSE)

→ 상대적 평균 오차, 평균오차와 비슷하지만 인스턴스의 개수가 아닌 실측 값의 평균으로 나눔

### 5. Coefficient of Determination ( $R^2$ )

→ 결정계수, 설명력으로도 불림

# Performance Index

## 1. Accuracy (ACC)

→ 정확도, 가장 보편적으로 사용

## 2. Precision & Recall

→ 정밀도 & 재현율, True라고 판단한 것 중 True인 비율 & 실제 True중 맞춘 True의 비율

## 3. F-Score (F1, F2, ...)

→ 정밀도와 재현율의 조화평균으로 F2,F3, 등으로 보기도 함

## 4. AUC (Area Under Curve)

→ ROC곡선 아래의 영역, 척도 불변

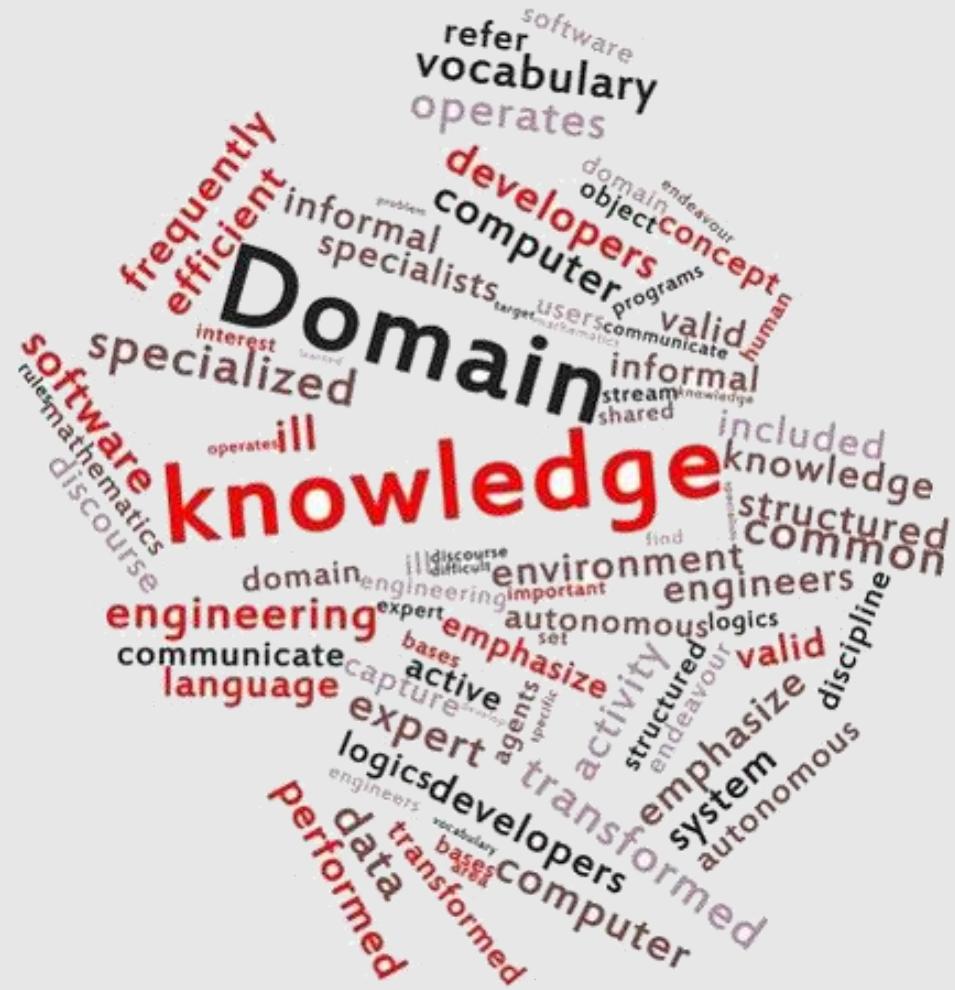
## 5. Log Loss(Cross-entropy)

→ 두 개 이상의 레이블에서만 정의,  $-\log P(y^t|y^p) = -(y^t \log(y^p) + (1 - y^t) \log(1 - y^p))$

## Look at the Big Picture

1. 목표를 비즈니스 용어로 정의합니다.
2. 이 솔루션은 어떻게 사용될 것인가?
3. (만약 있다면) 현재 솔루션이나 차선책은 무엇인가?
4. 어떤 문제라고 정의할 수 있나 (지도/비지도, 온라인/오프라인 등)?
  5. 성능을 어떻게 측정해야 하나?
6. 성능 지표가 비즈니스 목표에 연결되어 있나?
7. 비즈니스 목표에 도달하기 위해 필요한 최소한의 성능은 얼마인가?
8. 비슷한 문제가 있나? 이전의 방식이나 도구를 재사용 할 수 있나?
  9. 해당 분야의 전문가가 있나?
10. 수동으로 문제를 해결하는 방법은 무엇인가?
11. 여러분이 세운 가정을 나열합니다.
12. 가능하면 가정을 검정합니다.

# Performance Index



## Get the Data

1. 필요한 데이터와 양을 나열합니다.
2. 데이터를 얻을 수 있는 곳을 찾아 기록합니다.
3. 얼마나 많은 공간이 필요한지 확인합니다.
4. 법률상의 의무가 있는지 확인하고 필요하다면 인가를 받습니다.
5. 접근 권한을 획득합니다.
6. 작업 환경을 만듭니다.
7. 데이터를 수집합니다.
8. 데이터를 조작하기 편리한 형태로 변환합니다(데이터 자체는 바꾸지 않습니다).
9. 민감한 정보가 삭제되었거나 보호되었는지 검증합니다.
10. 데이터의 크기와 타입(시계열, 표본, 지리정보, 등)을 확인합니다.
11. 테스트 세트를 샘플링하여 따로 떼어놓고 절대 들여다보지 않습니다(데이터 염탐 금지!).

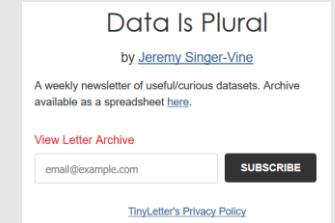
## Get the Data

1. 필요한 데이터와 양을 나열합니다.
2. 데이터를 얻을 수 있는 곳을 찾아 기록합니다.
3. 얼마나 많은 공간이 필요한지 확인합니다.
4. 법률상의 의무가 있는지 확인하고 필요하다면 인가를 받습니다.
5. 접근 권한을 획득합니다.
6. 작업 환경을 만듭니다.
7. 데이터를 수집합니다.
8. 데이터를 조작하기 편리한 형태로 변환합니다(데이터 자체는 바꾸지 않습니다).
9. 민감한 정보가 삭제되었거나 보호되었는지 검증합니다.
10. 데이터의 크기와 타입(시계열, 표본, 지리정보, 등)을 확인합니다.
11. 테스트 세트를 샘플링하여 따로 떼어놓고 절대 들여다보지 않습니다(데이터 염탐 금지!).

02 II. End-to-End Machine Learning Project

# Data Source Site

kaggle



⬆️ r/MachineLearning · Posted by u/ComfortablyNumb190 5 days ago  
116 Discussion [D] Where do you find datasets that you need?

Assuming you're looking for something specific and can't find it in publicly available repositories like UCI. Where would you get that dataset?

66 Comments Share Save Hide Report ... 95% Upvoted

What are your thoughts? Log in or Sign up

LOG IN SIGN UP

SORT BY BEST ▾

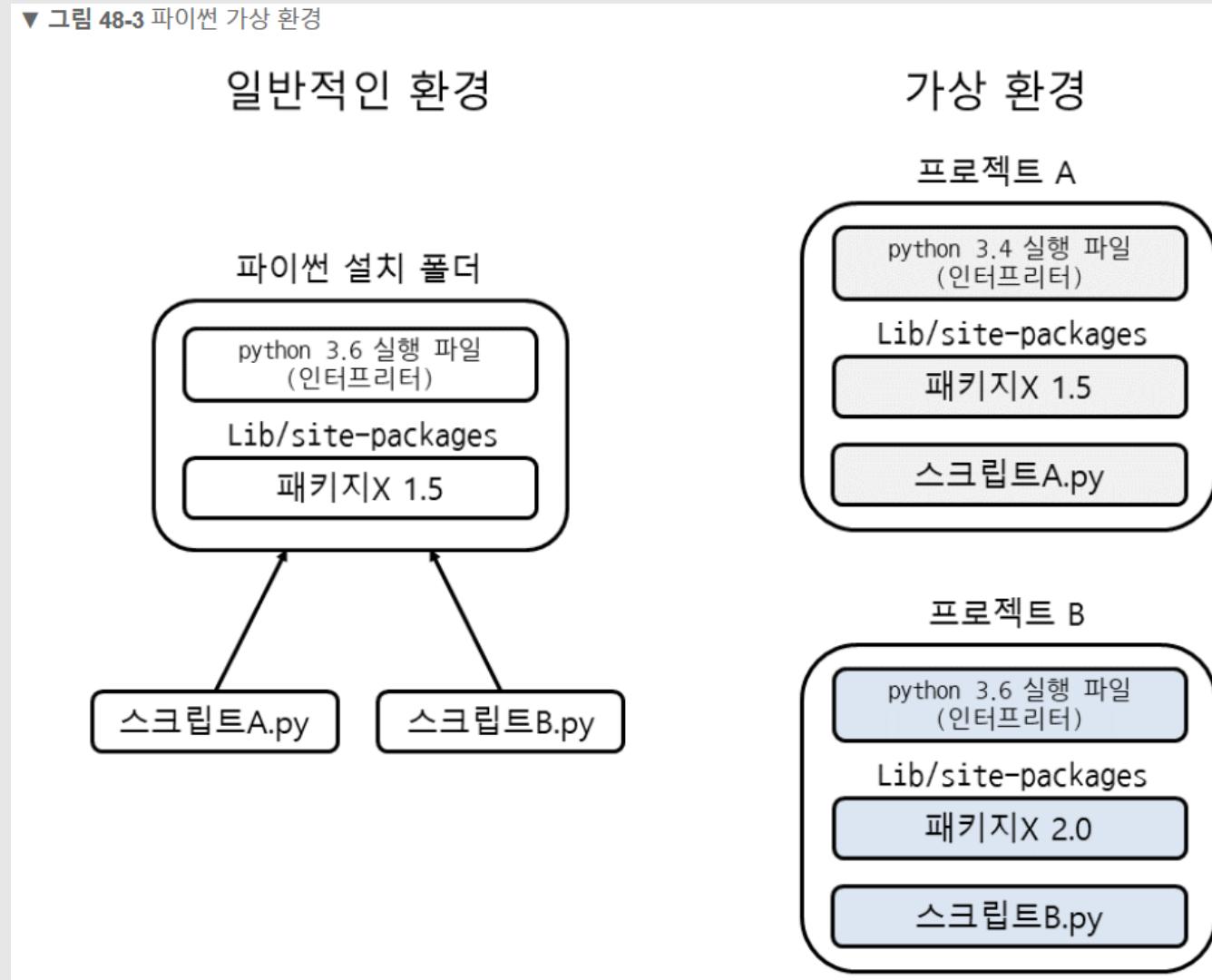
⬆️ tuttoweb 122 points · 5 days ago  
⬇️ Recently I found this sweet tool by google: <https://toolbox.google.com/datasetsearch>

Reply Share Report Save

02 II. End-to-End Machine Learning Project

# Virtual Environment

▼ 그림 48-3 파이썬 가상 환경



## Get the Data

1. 필요한 데이터와 양을 나열합니다.
2. 데이터를 얻을 수 있는 곳을 찾아 기록합니다.
3. 얼마나 많은 공간이 필요한지 확인합니다.
4. 법률상의 의무가 있는지 확인하고 필요하다면 인가를 받습니다.
5. 접근 권한을 획득합니다.
6. 작업 환경을 만듭니다.
7. 데이터를 수집합니다.
8. 데이터를 조작하기 편리한 형태로 변환합니다(데이터 자체는 바꾸지 않습니다).
9. 민감한 정보가 삭제되었거나 보호되었는지 검증합니다.
10. 데이터의 크기와 타입(시계열, 표본, 지리정보, 등)을 확인합니다.
11. 테스트 세트를 샘플링하여 따로 떼어놓고 절대 들여다보지 않습니다(데이터 염탐 금지!).

02

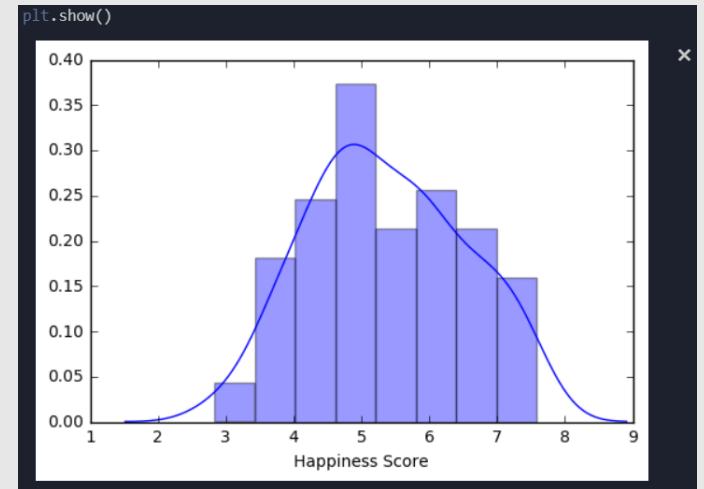
## II. End-to-End Machine Learning Project

# Virtual Environment

```
dati.head()
```

|   | Country     | Region         | Happiness Rank | Happiness Score | Standard Error | Economy (GDP per Capita) | Family  | Health (Life Expectancy) | Freedom | Trust (Government Corruption) | Generosity |
|---|-------------|----------------|----------------|-----------------|----------------|--------------------------|---------|--------------------------|---------|-------------------------------|------------|
| 0 | Switzerland | Western Europe | 1              | 7.587           | 0.03411        | 1.39651                  | 1.34951 | 0.94143                  | 0.66557 | 0.41978                       | 0.29678    |
| 1 | Iceland     | Western Europe | 2              | 7.561           | 0.04884        | 1.30232                  | 1.40223 | 0.94784                  | 0.62877 | 0.14145                       | 0.43630    |
| 2 | Denmark     | Western Europe | 3              | 7.527           | 0.03328        | 1.32548                  | 1.36058 | 0.87464                  | 0.64938 | 0.48357                       | 0.34139    |
| 3 | Norway      | Western Europe | 4              | 7.522           | 0.03880        | 1.45900                  | 1.33095 | 0.88521                  | 0.66973 | 0.36503                       | 0.34699    |
| 4 | Canada      | North America  | 5              | 7.427           | 0.03553        | 1.32629                  | 1.32261 | 0.90563                  | 0.63297 | 0.32957                       | 0.45811    |

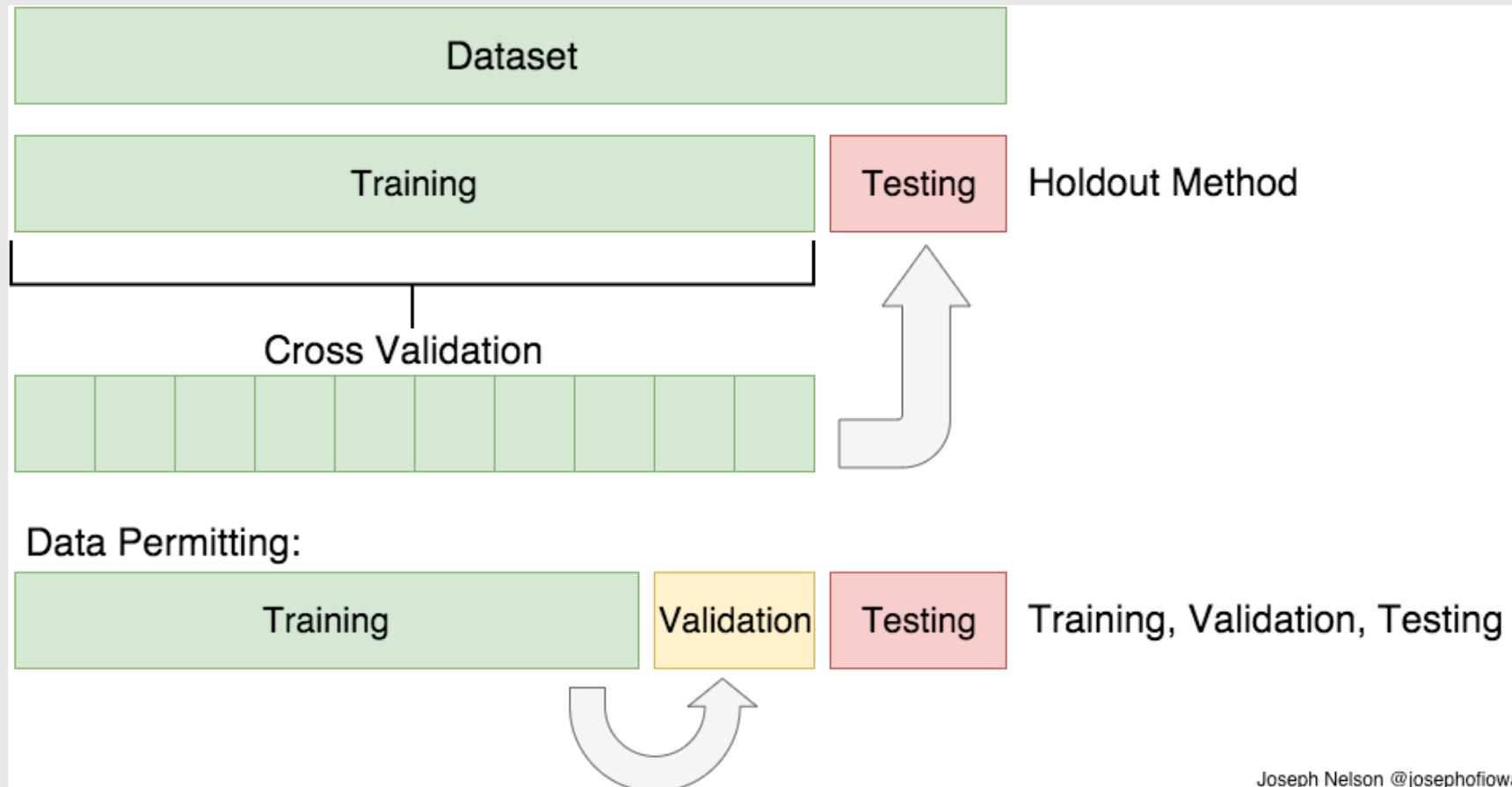
```
dati.info()  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 158 entries, 0 to 157  
Data columns (total 12 columns):  
Country          158 non-null object  
Region           158 non-null object  
Happiness Rank   158 non-null int64  
Happiness Score  158 non-null float64  
Standard Error   158 non-null float64  
Economy (GDP per Capita) 158 non-null float64  
Family            158 non-null float64  
Health (Life Expectancy) 158 non-null float64  
Freedom           158 non-null float64  
Trust (Government Corruption) 158 non-null float64  
Generosity         158 non-null float64  
Dystopia Residual 158 non-null float64  
dtypes: float64(9), int64(1), object(2)  
memory usage: 14.9+ KB
```



## Get the Data

1. 필요한 데이터와 양을 나열합니다.
2. 데이터를 얻을 수 있는 곳을 찾아 기록합니다.
3. 얼마나 많은 공간이 필요한지 확인합니다.
4. 법률상의 의무가 있는지 확인하고 필요하다면 인가를 받습니다.
5. 접근 권한을 획득합니다.
6. 작업 환경을 만듭니다.
7. 데이터를 수집합니다.
8. 데이터를 조작하기 편리한 형태로 변환합니다(데이터 자체는 바꾸지 않습니다).
9. 민감한 정보가 삭제되었거나 보호되었는지 검증합니다.
10. 데이터의 크기와 타입(시계열, 표본, 지리정보, 등)을 확인합니다.
11. 테스트 세트를 샘플링하여 따로 떼어놓고 절대 들여다보지 않습니다(데이터 염탐 금지!).

# Train, Test, Validation



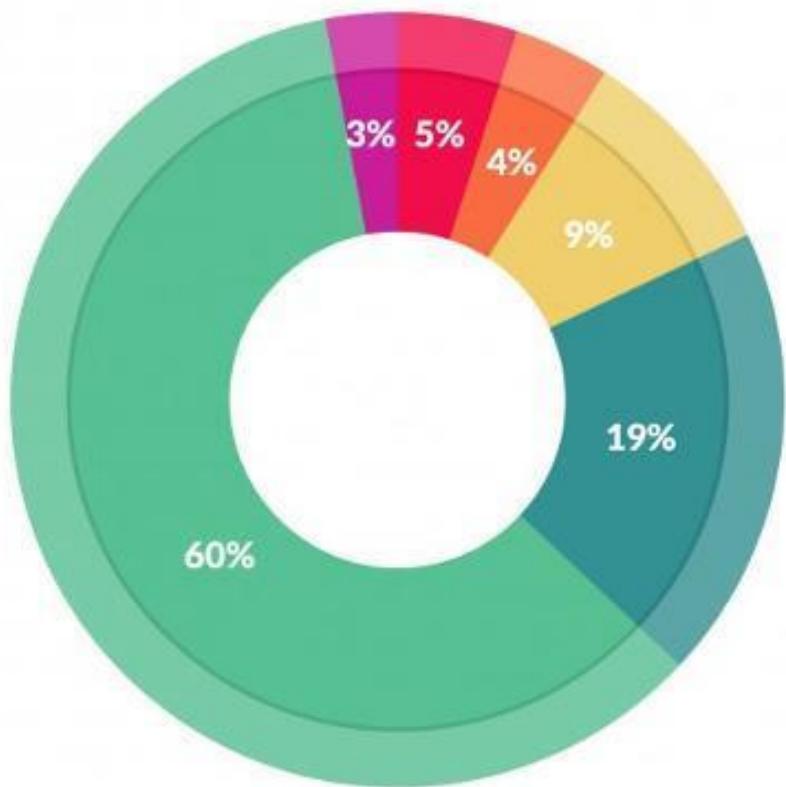
## Discover and Visualize the Data to Gain Insights

1. 데이터 탐색을 위해 복사본을 생성합니다.
2. 데이터 탐색 결과를 저장하기 위해 주피터 노트북을 만듭니다.
  3. 각 특성(Feature)의 특징을 조사합니다.
    - 이름, 타입, 누락된 값의 비율, 잡음의 정도와 비율, 분포 형태, 등-
  4. 지도 학습 작업이라면 타깃 특성을 구분합니다.
    5. 데이터를 시각화합니다.
    6. 특성 간의 상관관계를 조사합니다.
  7. 수동으로 문제를 해결할 수 있는 방법을 찾아봅니다.
  8. 적용이 가능한 변환을 찾습니다.
9. 추가로 유용한 데이터를 찾습니다 (있다면 ‘데이터를 수집합니다’로 돌아갑니다).
10. 조사한 것을 기록합니다.

## Discover and Visualize the Data to Gain Insights

1. 데이터 탐색을 위해 복사본을 생성합니다.
2. 데이터 탐색 결과를 저장하기 위해 주피터 노트북을 만듭니다.
  3. 각 특성(Feature)의 특징을 조사합니다.
    - 이름, 타입, 누락된 값의 비율, 잡음의 정도와 비율, 분포 형태, 등-
  4. 지도 학습 작업이라면 타깃 특성을 구분합니다.
    5. 데이터를 시각화합니다.
    6. 특성 간의 상관관계를 조사합니다.
  7. 수동으로 문제를 해결할 수 있는 방법을 찾아봅니다.
  8. 적용이 가능한 변환을 찾습니다.
9. 추가로 유용한 데이터를 찾습니다 (있다면 ‘데이터를 수집합니다’로 돌아갑니다).
10. 조사한 것을 기록합니다.

## Discover and Visualize the Data to Gain Insights



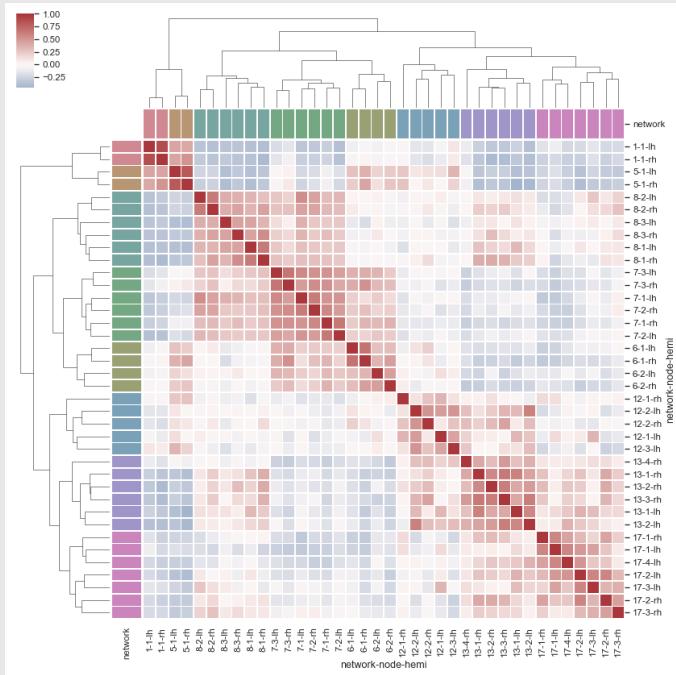
What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

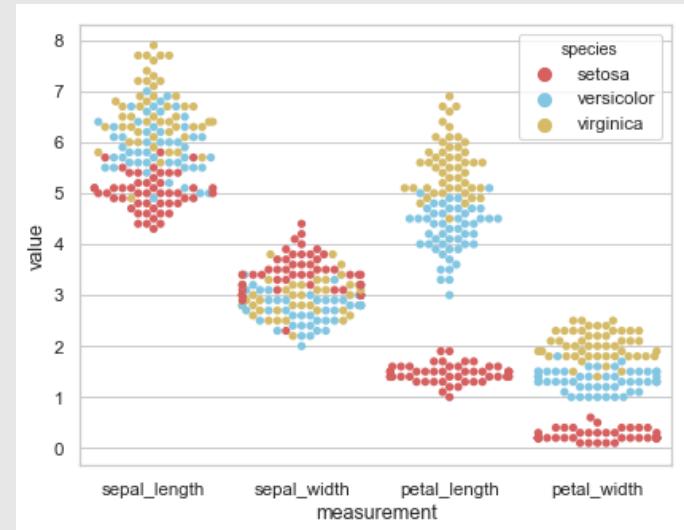
## Discover and Visualize the Data to Gain Insights

1. 데이터 탐색을 위해 복사본을 생성합니다.
2. 데이터 탐색 결과를 저장하기 위해 주피터 노트북을 만듭니다.
  3. 각 특성(Feature)의 특징을 조사합니다.
    - 이름, 타입, 누락된 값의 비율, 잡음의 정도와 비율, 분포 형태, 등-
  4. 지도 학습 작업이라면 타깃 특성을 구분합니다.
    5. 데이터를 시각화합니다.
    6. 특성 간의 상관관계를 조사합니다.
  7. 수동으로 문제를 해결할 수 있는 방법을 찾아봅니다.
  8. 적용이 가능한 변환을 찾습니다.
9. 추가로 유용한 데이터를 찾습니다 (있다면 ‘데이터를 수집합니다’로 돌아갑니다).
10. 조사한 것을 기록합니다.

# Data Visualize



# Cluster Map

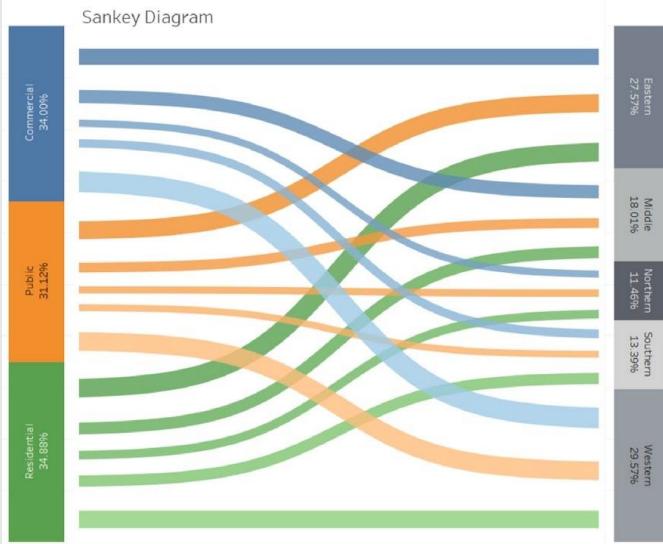


## Strip Plot (Swarm Plot)

## KDE Plot

02 II. End-to-End Machine Learning Project

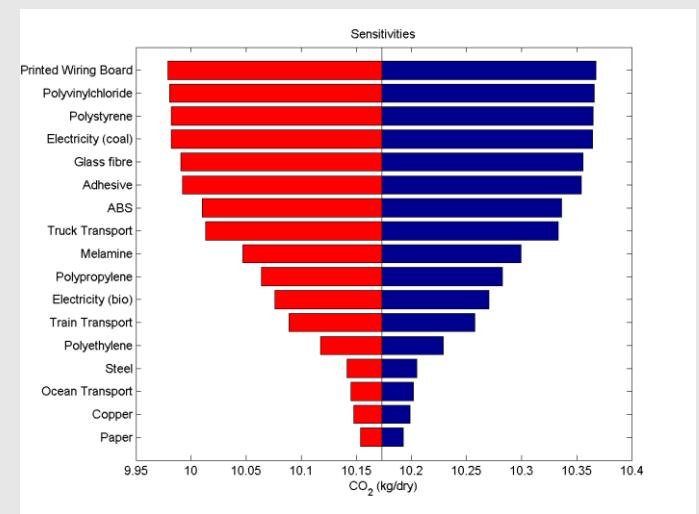
# Data Visualize



Sankey Diagram



Sunburst Plot



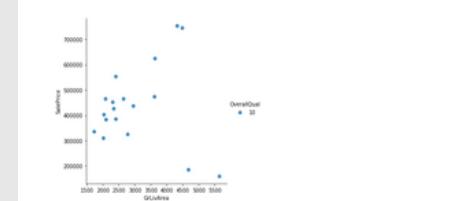
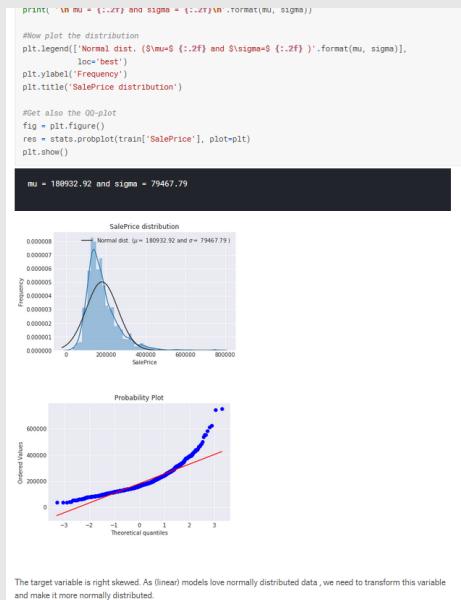
Tornado Plot

## Discover and Visualize the Data to Gain Insights

1. 데이터 탐색을 위해 복사본을 생성합니다.
2. 데이터 탐색 결과를 저장하기 위해 주피터 노트북을 만듭니다.
  3. 각 특성(Feature)의 특징을 조사합니다.
    - 이름, 타입, 누락된 값의 비율, 잡음의 정도와 비율, 분포 형태, 등-
  4. 지도 학습 작업이라면 타깃 특성을 구분합니다.
  5. 데이터를 시각화합니다.
  6. 특성 간의 상관관계를 조사합니다.
7. 수동으로 문제를 해결할 수 있는 방법을 찾아봅니다.
8. 적용이 가능한 변환을 찾습니다.
9. 추가로 유용한 데이터를 찾습니다 (있다면 ‘데이터를 수집합니다’로 돌아갑니다).
10. 조사한 것을 기록합니다.

## 02 II. End-to-End Machine Learning Project

# Data Insights



In [1]:  
Run outlier detection on OverallQual of 10  
From the chart above we can clearly see that there are a few outliers for quality 10 as well run the Local Outlier Factor to detect these outliers. (anything with a -1 is an outlier)

```
def detect_outlier():
    outlier_detect = explorer_data.In[explorer_data['OverallQual'] == 10]
    outlier_detect = outlier_detect[['SalePrice', 'GrLivArea']]
    outlier_detect['outlier'] = IsolationForest(n_neighbors=50).fit_predict(outlier_detect)
    print(outlier_detect)
```

|      | SalePrice | GrLivArea | outlier |
|------|-----------|-----------|---------|
| 55   | 458000    | 2145      | 1       |
| 125  | 340000    | 2295      | 1       |
| 224  | 336250    | 2392      | 1       |
| 349  | 426000    | 2332      | 1       |
| 440  | 555000    | 2402      | 1       |
| 515  | 402050    | 2028      | 1       |
| 523  | 184750    | 4676      | 1       |
| 585  | 325000    | 2775      | 1       |
| 591  | 451050    | 2396      | -1      |
| 691  | 755000    | 4316      | 1       |
| 725  | 315000    | 2054      | 1       |
| 994  | 337500    | 1715      | 1       |
| 1149 | 625000    | 3827      | 1       |

Serigne

Stacked Regressions : Top 4% on LeaderBoard

last run 9 months ago · IPython Notebook HTML · 110,086 views  
using data from House Prices: Advanced Regression Techniques · ...

1833 voters

Fork Notebook

<https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard>

## Prepare the Data for Machine Learning Algorithms

1. 데이터 정제
  - 이상치 수정 및 삭제, 데이터 Imputation
2. 특성 선택(선택사항)
3. 적절한 특성 공학 (Feature Engineering)
  - 연속 특성 이산화하기
  - 특성 분해하기
  - 가능한 특성 변환 추가하기
- 특성을 조합해 가능성 있는 새로운 특성 만들기
4. 특성 스케일 조정: 표준화 또는 정규화

## Prepare the Data for Machine Learning Algorithms

- 어떤 데이터셋에 대해서도 데이터 변환을 손쉽게 반복할 수 있습니다.
- 향후 프로젝트에 사용할 수 있는 변환 라이브러리를 점진적으로 구축하게 됩니다.
- 실제 시스템에서 알고리즘에 새 데이터를 주입하기 전에 변환시키는 데 이 함수를 사용할 수 있습니다.
  - 여러 가지 데이터 변환을 쉽게 시도해볼 수 있고 어떤 조합이 가장 좋은지 확인하는데 편리합니다.

## Prepare the Data for Machine Learning Algorithms

1. 데이터 정제
  - 이상치 수정 및 삭제, 데이터 Imputation
2. 특성 선택(선택사항)
3. 적절한 특성 공학 (Feature Engineering)
  - 연속 특성 이산화하기
  - 특성 분해하기
  - 가능한 특성 변환 추가하기
- 특성을 조합해 가능성 있는 새로운 특성 만들기
4. 특성 스케일 조정: 표준화 또는 정규화

## Prepare the Data for Machine Learning Algorithms

1. 데이터 정제
  - 이상치 수정 및 삭제, 데이터 Imputation
2. 특성 선택(선택사항)
3. 적절한 특성 공학 (Feature Engineering)
  - 연속 특성 이산화하기
  - 특성 분해하기
  - 가능한 특성 변환 추가하기
  - 특성을 조합해 가능성 있는 새로운 특성 만들기
4. 특성 스케일 조정: 표준화 또는 정규화

# Data Imputation



## Prepare the Data for Machine Learning Algorithms

1. 데이터 정제
  - 이상치 수정 및 삭제, 데이터 Imputation
2. 특성 선택(선택사항)
3. 적절한 특성 공학 (Feature Engineering)
  - 연속 특성 이산화하기
  - 특성 분해하기
  - 가능한 특성 변환 추가하기
  - 특성을 조합해 가능성 있는 새로운 특성 만들기
4. 특성 스케일 조정: 표준화 또는 정규화

## Feature Engineering

### Feature Engineering이란?

- Feature Engineering은 머신러닝 알고리즘을 작동하기 위해 데이터에 대한 도메인 지식을 활용하여 특징(Feature)를 만들어내는 과정입니다.
- 다른 정의를 살펴보면 머신러닝 모델을 위한 데이터 테이블의 Column(특징)을 생성하거나 선택하는 작업을 의미한다고 합니다.
- 간단히 정리하면 모델의 성능을 높이기 위해 모델에 입력할 데이터를 만들기 위해 주어진 초기 데이터로부터 특징을 가공하고 생성하는 전체 과정을 의미합니다.
- Feature Engineering은 모델 성능에 미치는 영향이 크기 때문에 머신러닝 응용에 있어서 굉장히 중요한 단계이며 전문성과 시간과 비용이 많이 드는 작업입니다.

# Feature Engineering

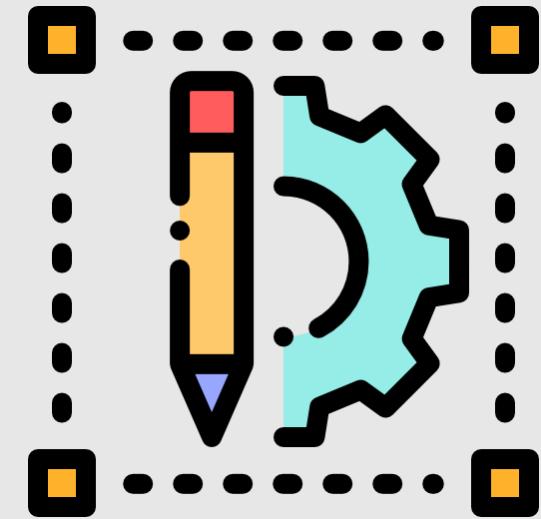
## 방법적인 측면



특징 선택  
(Feature Selection)



차원 축소  
(Dimension Reduction)



특징 구축  
(Feature Generation)

## Feature Engineering

### Feature Engineering Process

1. Brainstorming or Testing features
2. Deciding what features to create
  3. Creating features
4. Checking how the features work with your model
5. Improving your features if needed
6. Go back to brainstorming/creating more features until the work is done.

# Feature Engineering

관점에 따른 분류



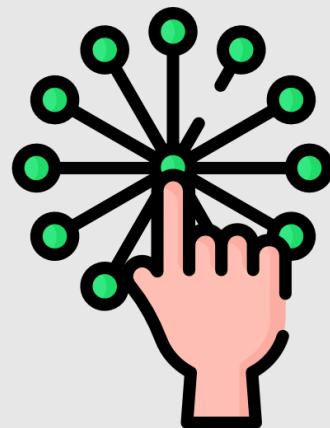
Business Driven  
Features



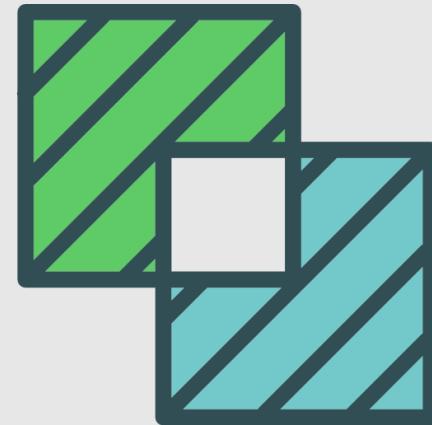
Data Driven  
Features

# Feature Engineering

방법에 따른 분류



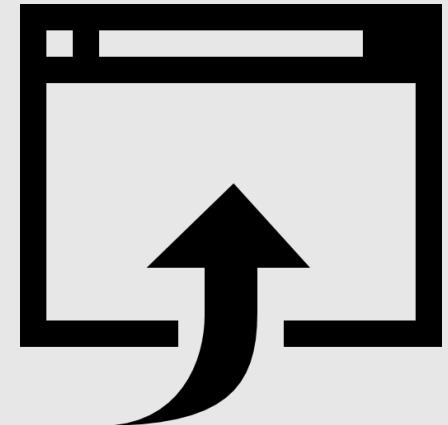
Indicator Variables



Interaction Features



Feature Representation



External Data

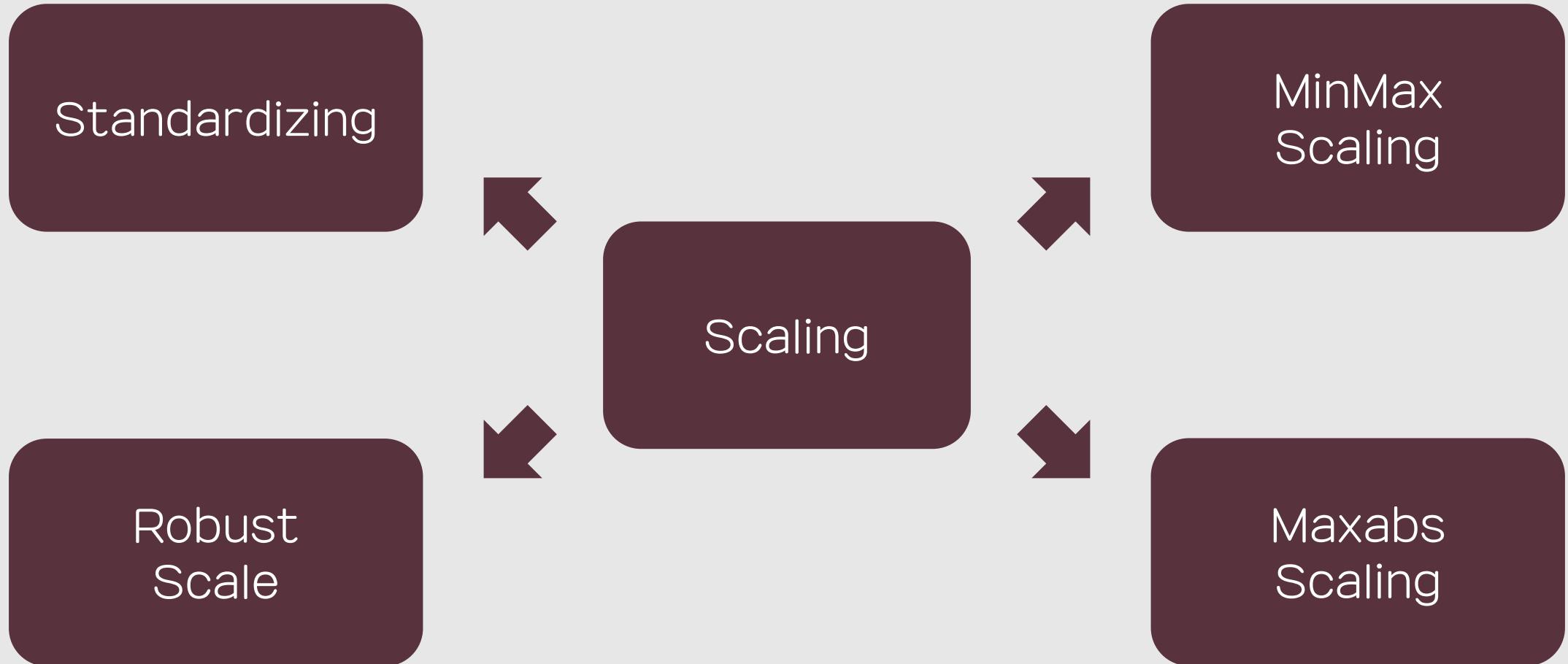
## Feature Engineering

- **에러 분석(Error Analysis - Post-Modeling)** : 모델을 통해 나온 결과를 바탕으로 특징을 만드는 방법입니다. 일반적으로 데이터 사이언스의 프로세스가 반복을 기반으로 모델의 성능을 높이기 때문에 당연하다고 생각하실 수도 있을 것 같습니다. 그래서 아래에 보다 구체적으로 구분한 에러 분석을 통해 특징을 만드는 방법을 소개해드리겠습니다.
  - **Start with larger errors** : 모델을 통해 나온 모든 값을 확인하기 보다 '에러(Error)'값이 큰 feature부터 확인하는 방법입니다.
  - **Segment By classes** : 평균 에러(Error)값을 기준으로 Segment를 나누어 비교하면서 분석하는 방법입니다.
  - **Unsupervised clustering** : 패턴을 발견하는데 어려움이 있을 경우 Unsupervised(비지도) 학습인 clustering 알고리즘을 사용하여 분류되지 않은 값들을 확인하는 방법입니다. 여기서 clustering을 클래스로 분류하는 것이 아니라 에러(Error)의 원인을 찾는 방법으로 사용해야 한다는 점을 주의해야 합니다.
  - **Ask colleagues or domain experts** : 데이터를 통해서 발견할 수 없다면 도메인(분야) 전문가의 도움을 통해 에러(Error)의 원인을 찾아낼 수도 있습니다.

## Prepare the Data for Machine Learning Algorithms

1. 데이터 정제
  - 이상치 수정 및 삭제, 데이터 Imputation
2. 특성 선택(선택사항)
3. 적절한 특성 공학 (Feature Engineering)
  - 연속 특성 이산화하기
  - 특성 분해하기
  - 가능한 특성 변환 추가하기
  - 특성을 조합해 가능성 있는 새로운 특성 만들기
4. 특성 스케일 조정: 표준화 또는 정규화

# Scaling



## Scaling VS Normalization

### Scaling? Normalization?

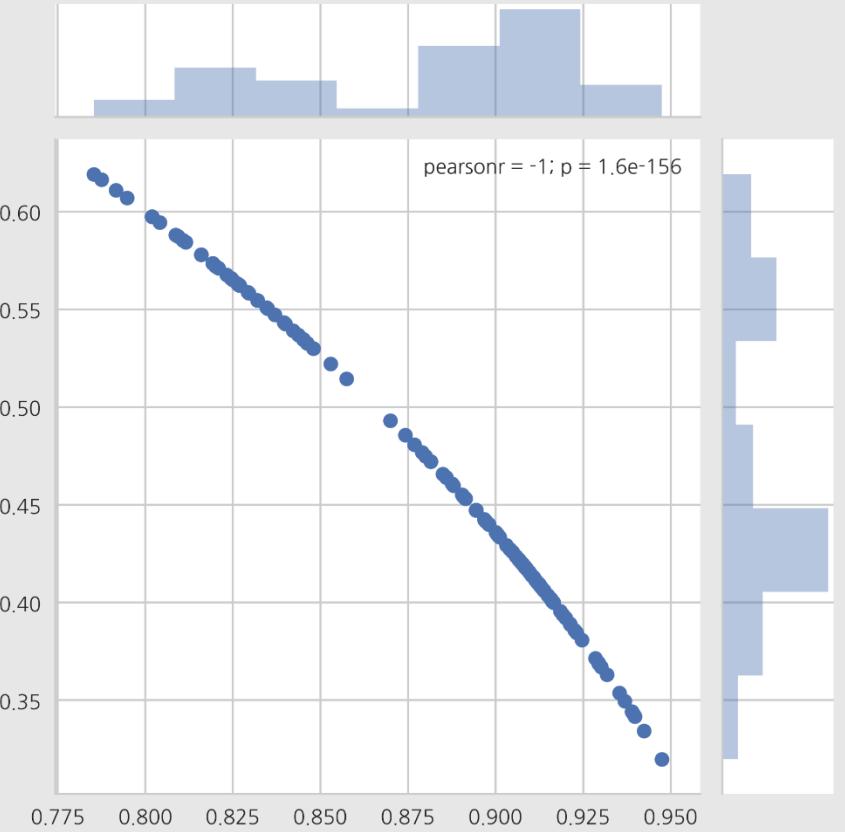
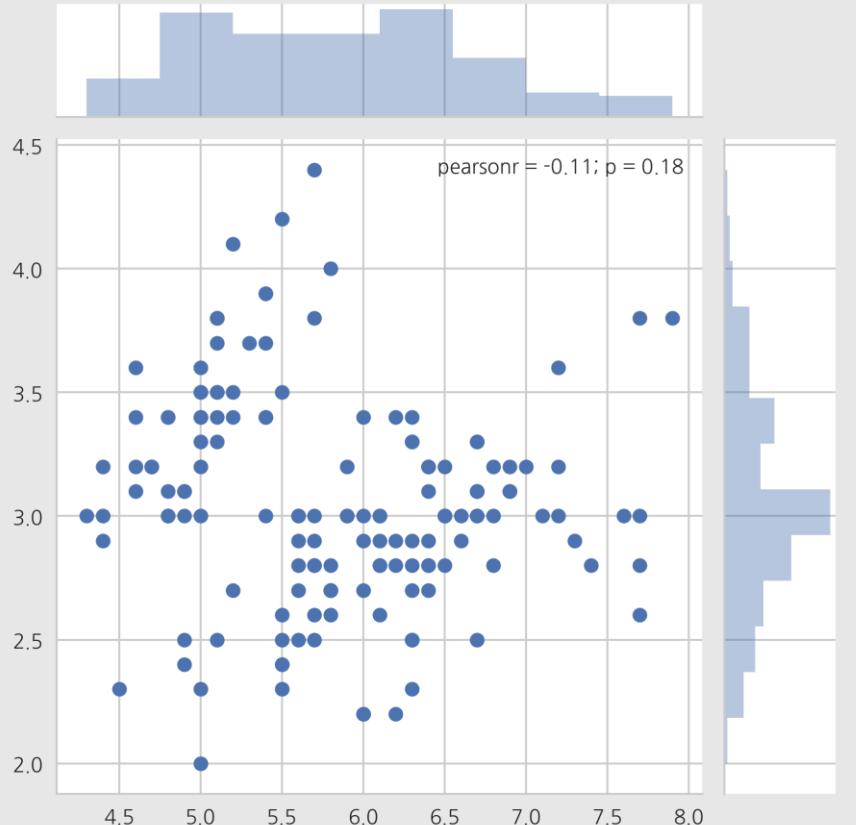
- 정규화는 개별 데이터의 크기를 모두 같게 만들기 위한 변환
- 개별 데이터에 대하여 서로 다른 변환 계수가 적용
  - 다차원 독립 변수 벡터가 있을 때  
각 벡터 원소들의 상대적 크기만 중요한 경우에 사용

```
original x:  
[[ -20. -2.]  
[ -19. -1.]  
[ -18.  0.]  
[ -17.  1.]  
[ -16.  2.]]  
scale:  
[[ -1.41421356 -1.41421356]  
[ -0.70710678 -0.70710678]  
[  0.          0.        ]  
[  0.70710678  0.70710678]  
[  1.41421356  1.41421356]]  
norms (scale)  
[2. 1. 0. 1. 2.]
```

```
normalize:  
[[ -0.99503719 -0.09950372]  
[ -0.99861783 -0.05255883]  
[ -1.          0.        ]  
[ -0.99827437  0.05872202]  
[ -0.99227788  0.12403473]]  
norms (normalize)  
[1. 1. 1. 1. 1.]
```

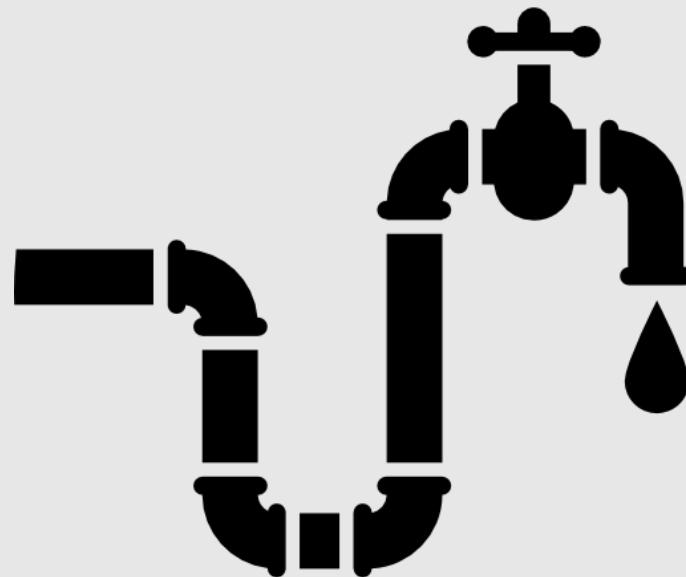
02 II. End-to-End Machine Learning Project

# Normalization



## Pipeline

프로세서로 가는 명령어들의 움직임 또는 명령어를 수행하기 위해  
프로세서에 의해 취해진 산술적인 단계가 연속적이고 겹치는 것을 의미



## Select and Train a Model

- 데이터가 너무 크면 여러 가지 모델을 일정 시간 안에 훈련시킬 수 있도록 데이터를 샘플링하여 작은 훈련세트를 만드는 것이 좋습니다.

- 여러 종류의 모델을 기본 매개변수를 사용해 신속하게 많이 훈련시켜 봅니다.
  - 성능을 측정하고 비교합니다.

(각 모델에서 N-교차검증을 통해 N개의 폴드의 성능에 대한 평균과 표준편차를 계산합니다.)

- 각 알고리즘에서 가장 두드러진 변수를 분석합니다.
  - 모델이 만드는 에러의 종류를 분석합니다.

(이 에러를 피하기 위해서 사람이 사용해야 하는 데이터는?)

- 간단한 특성 선택과 특성 공학 단계를 거칩니다.
- 이전 다섯 단계를 한 번이나 두 번 빠르게 반복합니다.
- 다른 종류의 에러를 만드는 모델을 중심으로 가장 가능성성이 높은 모델을 세 개에서 다섯 개 정도로 추립니다.

## Select and Train a Model

- 데이터가 너무 크면 여러 가지 모델을 일정 시간 안에 훈련시킬 수 있도록 데이터를 샘플링하여 작은 훈련세트를 만드는 것이 좋습니다.

- 여러 종류의 모델을 기본 매개변수를 사용해 신속하게 많이 훈련시켜 봅니다.
  - 성능을 측정하고 비교합니다.

(각 모델에서 N-교차검증을 통해 N개의 폴드의 성능에 대한 평균과 표준편차를 계산합니다.)

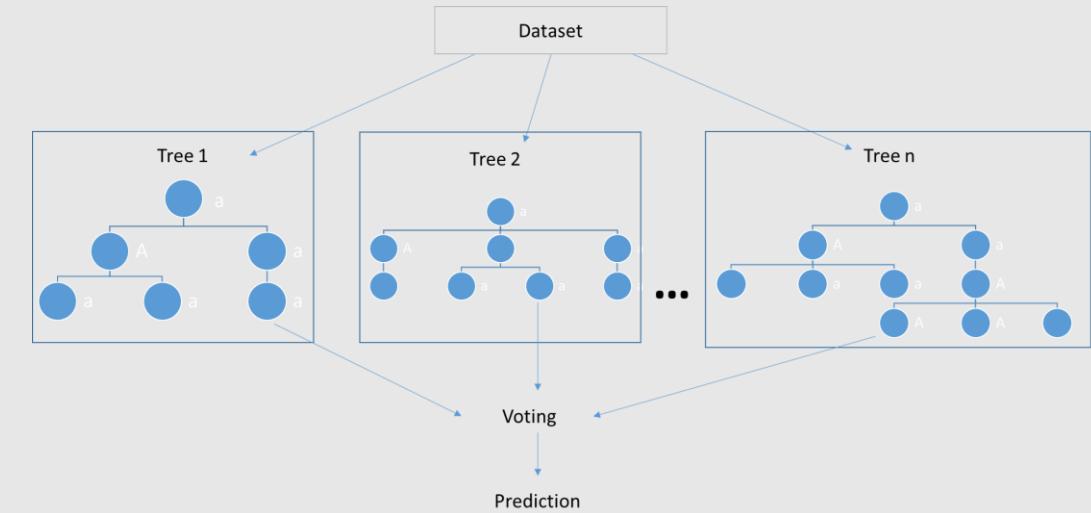
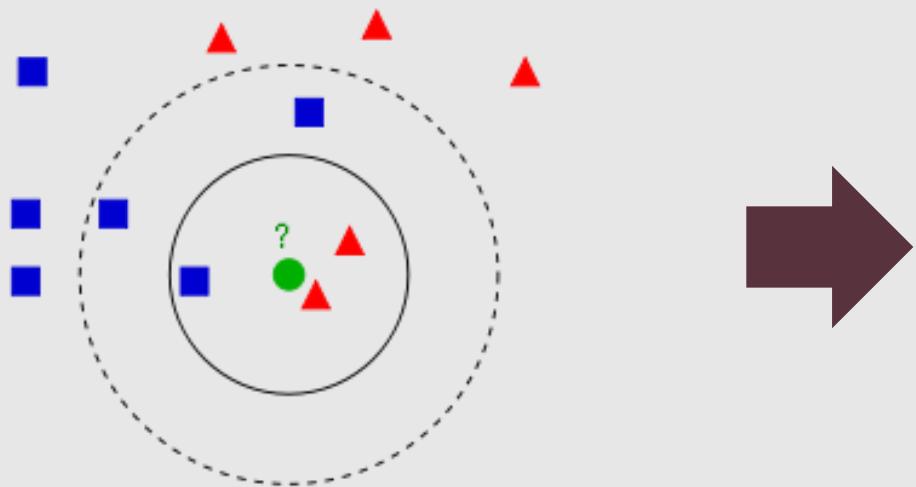
- 각 알고리즘에서 가장 두드러진 변수를 분석합니다.
  - 모델이 만드는 에러의 종류를 분석합니다.

(이 에러를 피하기 위해서 사람이 사용해야 하는 데이터는?)

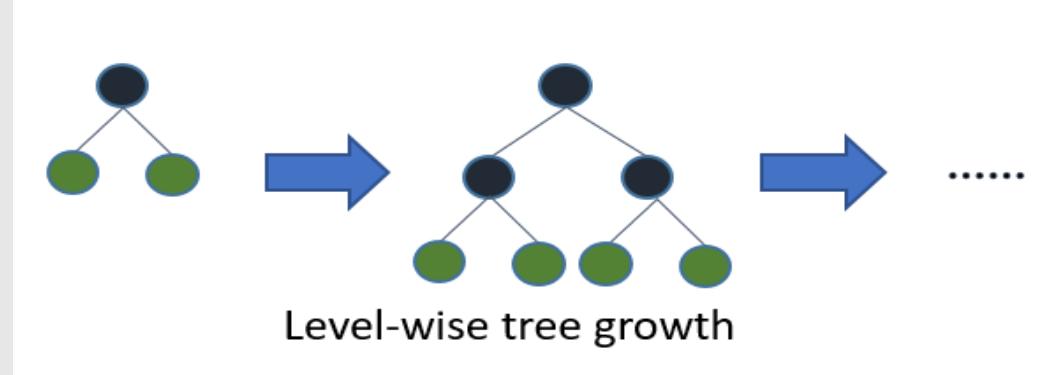
- 간단한 특성 선택과 특성 공학 단계를 거칩니다.
- 이전 다섯 단계를 한 번이나 두 번 빠르게 반복합니다.
- 다른 종류의 에러를 만드는 모델을 중심으로 가장 가능성성이 높은 모델을 세 개에서 다섯 개 정도로 추립니다.

02 II. End-to-End Machine Learning Project

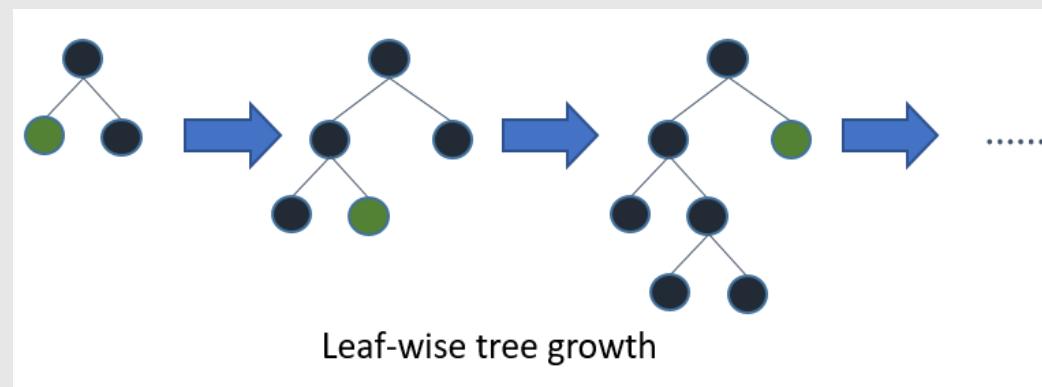
# Select and Train a Model



# XGBoost & LightGBM



XGBoost



LGBM

## Select and Train a Model

- 데이터가 너무 크면 여러 가지 모델을 일정 시간 안에 훈련시킬 수 있도록 데이터를 샘플링하여 작은 훈련세트를 만드는 것이 좋습니다.

- 여러 종류의 모델을 기본 매개변수를 사용해 신속하게 많이 훈련시켜 봅니다.
  - 성능을 측정하고 비교합니다.

(각 모델에서 N-교차검증을 통해 N개의 폴드의 성능에 대한 평균과 표준편차를 계산합니다.)

- 각 알고리즘에서 가장 두드러진 변수를 분석합니다.
  - 모델이 만드는 에러의 종류를 분석합니다.

(이 에러를 피하기 위해서 사람이 사용해야 하는 데이터는?)

- 간단한 특성 선택과 특성 공학 단계를 거칩니다.

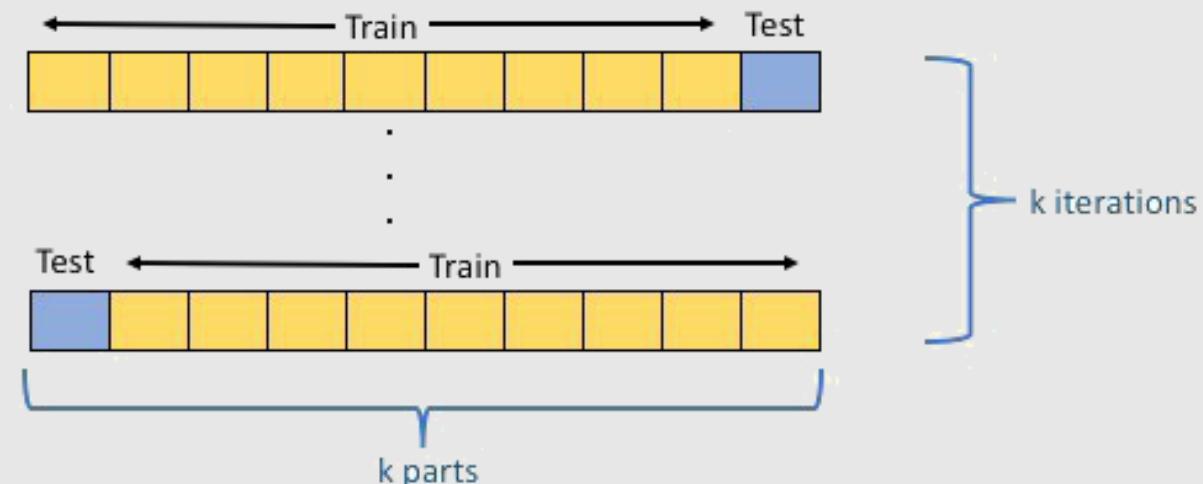
- 이전 다섯 단계를 한 번이나 두 번 빠르게 반복합니다.

- 다른 종류의 에러를 만드는 모델을 중심으로 가장 가능성성이 높은 모델을 세 개에서 다섯 개 정도로 추립니다.

# XGBoost & LightGBM

## K Folds Cross Validation Method

1. Divide the sample data into  $k$  parts.
2. Use  $k-1$  of the parts for training, and 1 for testing.
3. Repeat the procedure  $k$  times, rotating the test set.
4. Determine an expected performance metric (mean square error, misclassification error rate, confidence interval, or other appropriate metric) based on the results across the iterations



## Fine-Tune Your Model

- 이 단계에서는 가능한 한 많은 데이터를 사용하는 것이 좋습니다. 특히 세부 튜닝의 마지막 단계로 갈수록 그렇습니다.
- 언제나 그렇듯이 할 수 있다면 자동화하도록 합니다.
  1. 교차검증을 사용해 정밀 튜닝을 합니다.

(하이퍼 파라미터를 사용해 데이터 변환을 선택하세요. 특히 확신이 없는 경우에 이렇게 해야 합니다.)

(탐색할 하이퍼 파리미터의 값이 매우 적지 않다면 그리드 서치보다 랜덤 서치를 사용하세요. 훈련 시간이 오래 걸린다면 베이지안 최적화 방법을 사용하는 것이 좋습니다.)

2. 양상을 방법을 시도해보세요. 최고의 모델들을 연결하면 종종 개별 모델을 실행하는 것 보다 더 성능이 좋습니다.

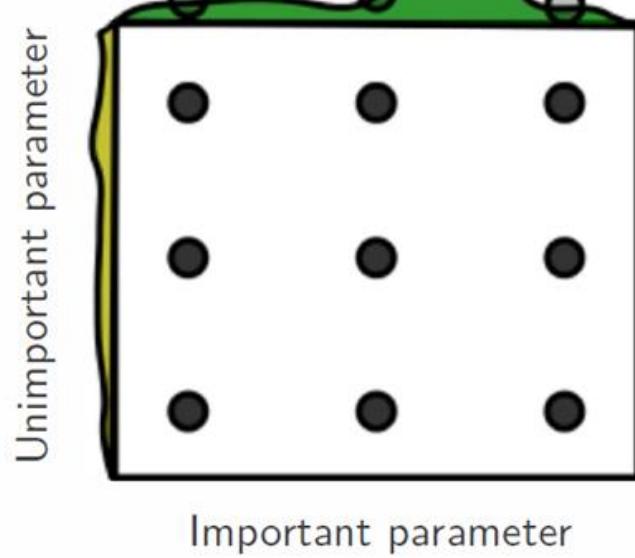
3. 최종모델에 확신이 선 후 일반화 오차를 추정하기 위해 테스트 세트에서 성능을 측정합니다.

## Fine-Tune Your Model

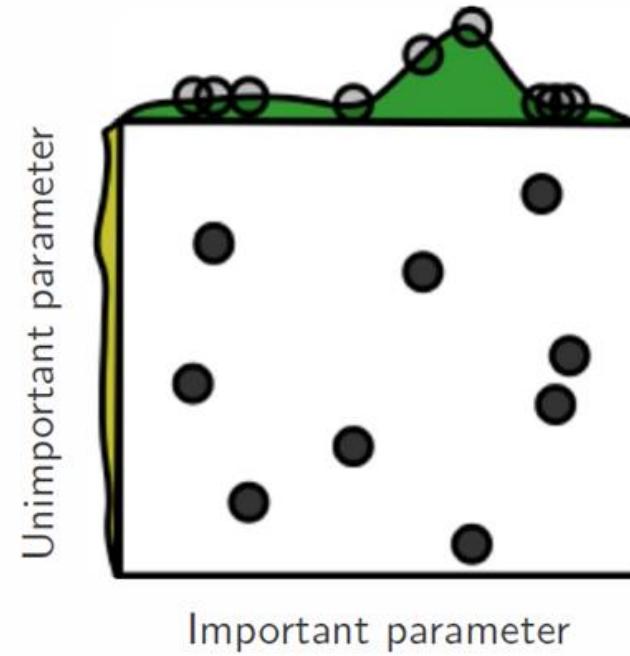
- 이 단계에서는 가능한 한 많은 데이터를 사용하는 것이 좋습니다.  
특히 세부 튜닝의 마지막 단계로 갈수록 그렇습니다.
- 언제나 그렇듯이 할 수 있다면 자동화하도록 합니다.
  1. 교차검증을 사용해 정밀 튜닝을 합니다.  
(하이퍼 파라미터를 사용해 데이터 변환을 선택하세요.  
특히 확신이 없는 경우에 이렇게 해야 합니다.)  
(탐색할 하이퍼 파라미터의 값이 매우 적지 않다면  
그리드 서치보다 랜덤 서치를 사용하세요.  
훈련 시간이 오래 걸린다면 베이지안 최적화 방법을 사용하는 것이 좋습니다.)
  2. 양상블 방법을 시도해보세요. 최고의 모델들을 연결하면 종종 개별 모델을 실행하는  
것 보다 더 성능이 좋습니다.
  3. 최종모델에 확신이 선 후 일반화 오차를 추정하기 위해 테스트 세트에서 성능을 측정합니다.

# Grid Search & Random Search

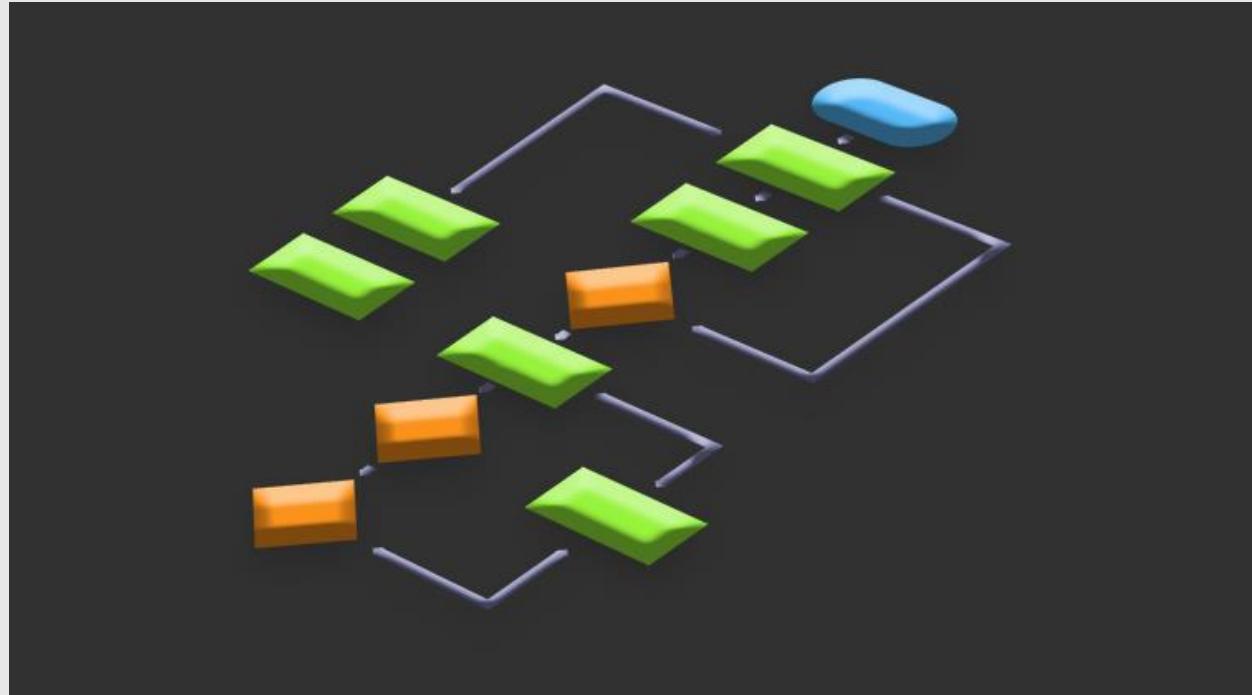
Grid Layout



Random Layout



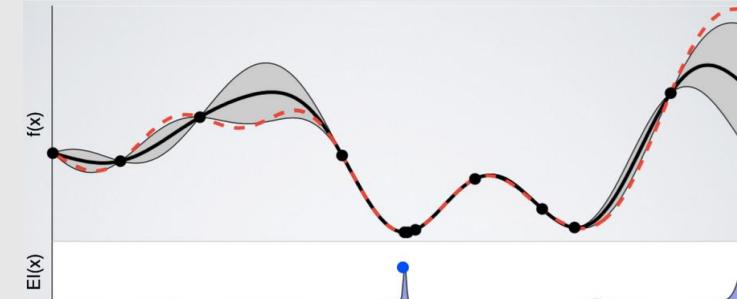
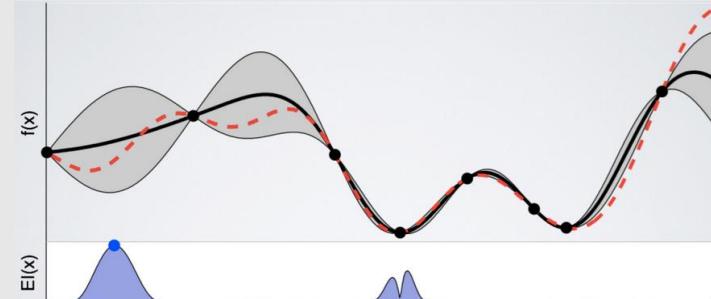
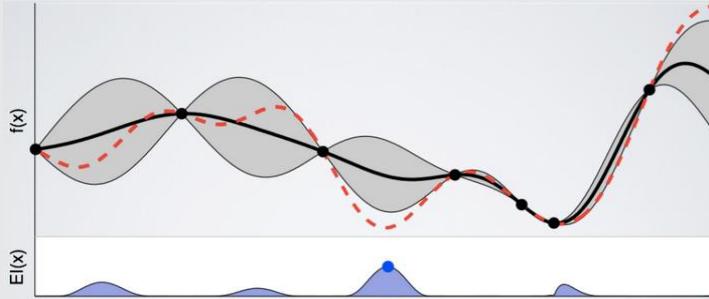
## Algorithm Class Recommend



MIT 6006

<https://www.edwith.org/introalgorithm>

# Bayesian Optimization



앞서 살펴본 3가지 방식이 좀 효율적이지 못한 감이 있다. Bayesian optimization의 기본 원리가 prior knowledge를 활용하는데 있으므로, 현재까지의 실험 결과를 바탕으로 통계적인 모델을 만들고, 그것을 바탕으로 다음 탐색을 해야 할 방향을 효과적으로 정하자는 것이 이 방법의 핵심이다. 유명한 논문으로는 Adams 등이 쓴 “Practical Bayesian optimization of machine learning algorithms” 가 있다. Bayesian optimization 방법을 사용하면 Random search나 Grid search를 사용하는 것에 비해 좀 더 짧은 시간에 최적 값을 찾아내는 경향이 있다.

(출처: [http://www.hellot.net/new\\_hellot/magazine/magazine\\_read.html?code=205&sub=001&idx=41604](http://www.hellot.net/new_hellot/magazine/magazine_read.html?code=205&sub=001&idx=41604))

## Fine-Tune Your Model

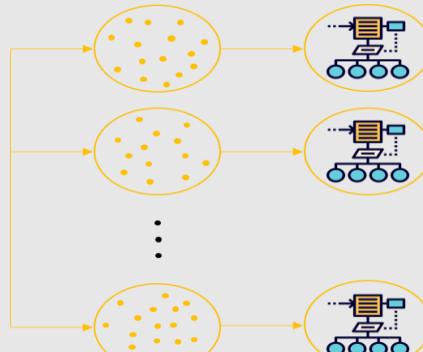
- 이 단계에서는 가능한 한 많은 데이터를 사용하는 것이 좋습니다.  
특히 세부 튜닝의 마지막 단계로 갈수록 그렇습니다.
  - 언제나 그렇듯이 할 수 있다면 자동화하도록 합니다.
    1. 교차검증을 사용해 정밀 튜닝을 합니다.  
(하이퍼 파라미터를 사용해 데이터 변환을 선택하세요.  
특히 확신이 없는 경우에 이렇게 해야 합니다.)  
(탐색할 하이퍼 파라미터의 값이 매우 적지 않다면  
그리드 서치보다 랜덤 서치를 사용하세요.  
훈련 시간이 오래 걸린다면 베이지안 최적화 방법을 사용하는 것이 좋습니다.)
2. 양상을 방법을 시도해보세요. 최고의 모델들을 연결하면 종종 개별 모델을 실행하는 것 보다 더 성능이 좋습니다.
3. 최종모델에 확신이 선 후 일반화 오차를 추정하기 위해 테스트 세트에서 성능을 측정합니다.

## Ensemble Method

### Bagging (배깅)

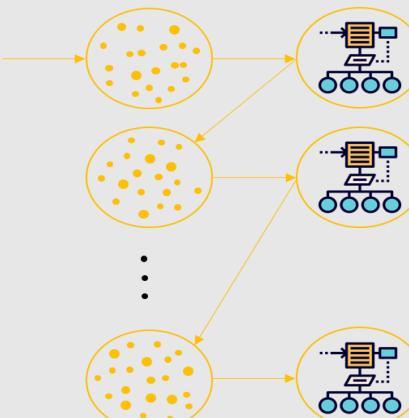
(=Bootstrap Aggregating)

주어진 데이터에 대해  
여러 개의 부트스트랩 자료를 생성하  
고 각 부트스트랩 자료를 모델링 한 후  
결합하는 방식



### Boosting (부스팅)

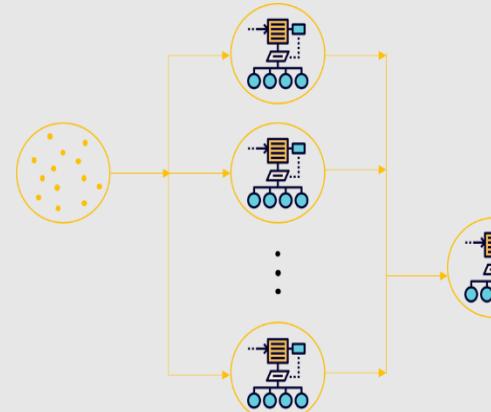
간단한 모형을 결합하는 방법으로  
일반적으로 잘못 분류된 개체들에게  
관심을 가지고 이들을 더 잘 만들도록  
새로운 규칙을 생성하는 방식



### Stacking (스태킹)

서로 다른 모델들을  
조합하여 최고의 성능을 내는 모델을  
생성하는 방식

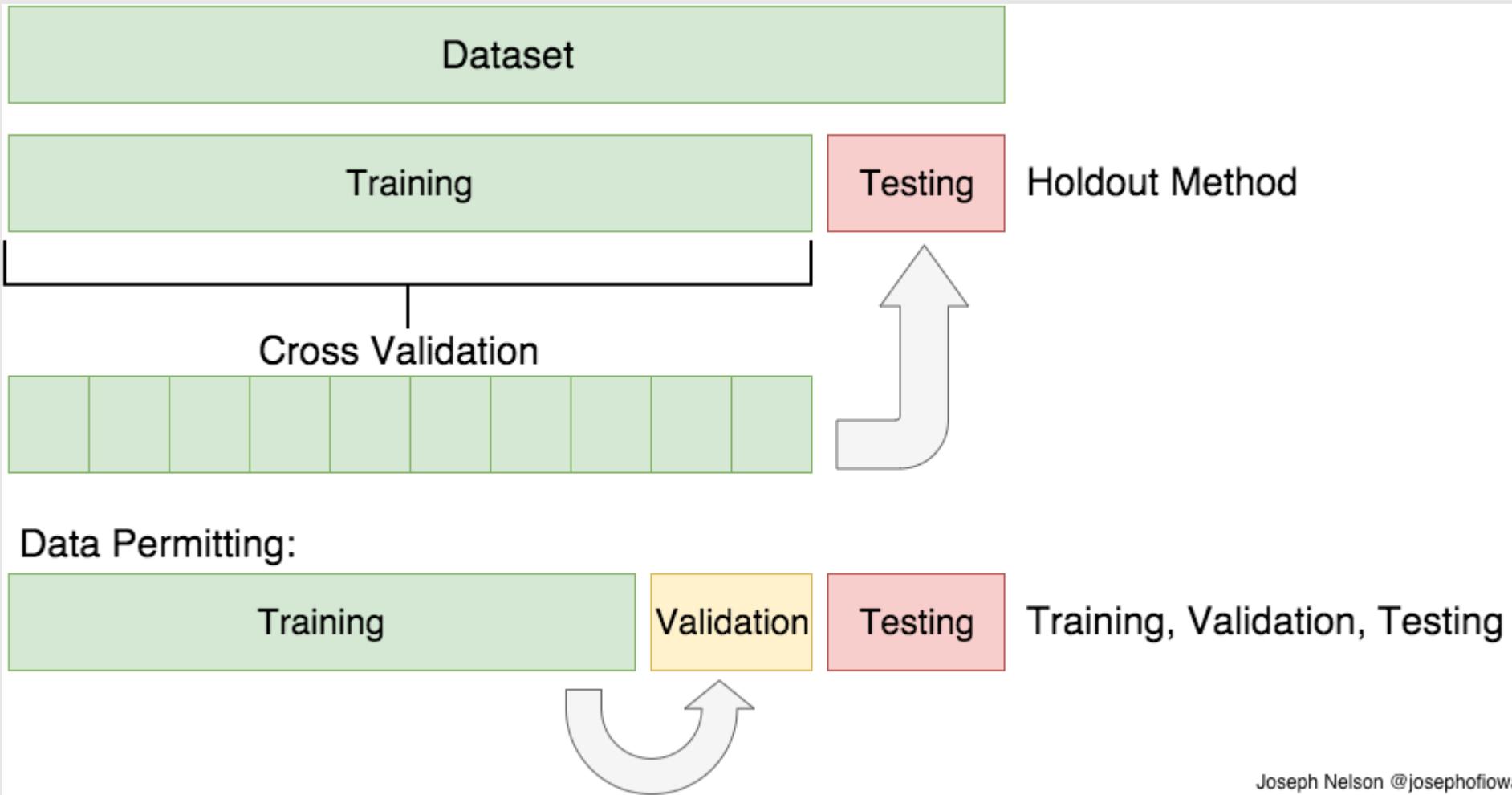
각 모델들의 장점을 취하고  
단점을 보완하는 것을 목적으로 함



## Fine-Tune Your Model

- 이 단계에서는 가능한 한 많은 데이터를 사용하는 것이 좋습니다.  
특히 세부 튜닝의 마지막 단계로 갈수록 그렇습니다.
- 언제나 그렇듯이 할 수 있다면 자동화하도록 합니다.
  1. 교차검증을 사용해 정밀 튜닝을 합니다.  
(하이퍼 파라미터를 사용해 데이터 변환을 선택하세요.  
특히 확신이 없는 경우에 이렇게 해야 합니다.)  
(탐색할 하이퍼 파라미터의 값이 매우 적지 않다면  
그리드 서치보다 랜덤 서치를 사용하세요.  
훈련 시간이 오래 걸린다면 베이지안 최적화 방법을 사용하는 것이 좋습니다.)
  2. 양상을 방법을 시도해보세요. 최고의 모델들을 연결하면 종종 개별 모델을 실행하는  
것 보다 더 성능이 좋습니다.
  3. 최종모델에 확신이 선 후 일반화 오차를 추정하기 위해 테스트 세트에서 성능을 측정합니다.

# Ensemble Method



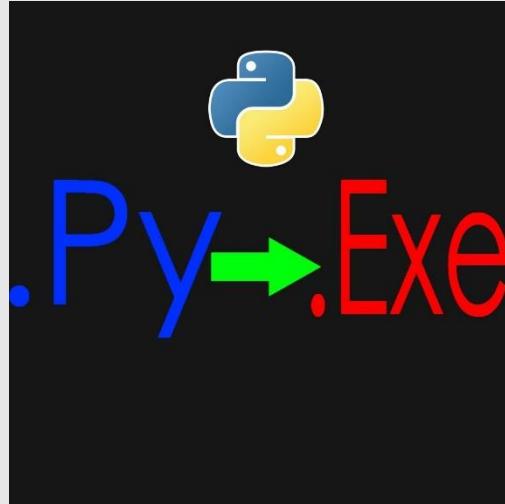
## Present a Solution

1. 지금까지의 작업을 문서화합니다.
2. 멋진 발표 자료를 만듭니다.
3. 이 솔루션이 어떻게 비즈니스 목표를 달성하는지 설명하세요.
4. 작업 과정에서 알게 된 흥미로운 점을 잊지 말고 설명하세요.  
(성공한 것과 그렇지 못한 것을 설명합니다.)  
(우리가 세운 가정과 시스템의 제약을 나열합니다.)
5. 멋진 그래프나 기억하기 쉬운 문장으로 핵심 내용을 전달하세요.

## Present a Solution

1. 서비스에 투입하기 위해 솔루션을 준비합니다.  
(실제 입력 데이터 연결, 단위 테스트 작성, 등)
2. 시스템의 서비스 성능을 일정한 간격으로 확인하고 성능이 감소됐을 때  
알림을 받기 위해 모니터링 코드를 작성합니다.  
(아주 느리게 감소되는 현상을 주의하세요. 데이터가 변화함에 따라 모델이 점차  
구식이 되는 경향이 있습니다.)  
(성능 측정에 사람의 개입이 필요할지 모릅니다.)  
(입력 데이터의 품질도 모니터링 합니다. 온라인 학습에서 특히 중요합니다.)
3. 정기적으로 새로운 데이터에서 모델을 다시 훈련시킵니다.  
(가능한 한 자동화 합니다.)

## Present a Solution



- py2exe가 가장 대표적
  - Freeze도 자주 쓰임
- Linux환경에서는 바로 bat으로 만들어서 사용 가능하나,  
그 Linux환경에 라이브러리 설치가 강요된다.



Ideas worth spreading  
- TED Talks

고생하셨습니다.