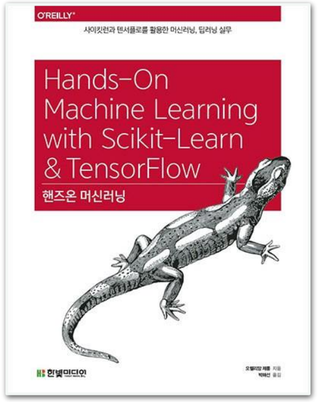



# 핸즈온 머신러닝

## 1장. 한눈에 보는 머신러닝

박해선(옮긴이) + alpha

# 한눈에 보는 머신러닝

- 머신러닝이란 무엇일까요?
- 기계가 배운다는 것은 정확히 무엇을 의미하는 걸까요?
- 머신러닝이 왜 필요한가요?

이 장에서는 머신러닝의 시스템 종류, 주요 도전 과제, 평가와 튜닝 등 머신러닝 전체에  대한 그림을 그려봅니다.

# 머신러닝이란

- 현업에서 본 머신러닝 : https://www.youtube.com/watch?v=aF03asAmQbY

- “머신러닝”은 명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구 분야다. (아서 사무엘Arthur Samuel, 1959)

- 어떤 작업 T에 대한 컴퓨터 프로그램의 성능을 P로 측정했을 때 경험 E로 인해 성능이 향상됐다면, 이 컴퓨터 프로그램은 작업 T와 성능 측정 P에 대해 경험 E로 학습한 것이다. (톰 미첼Tom Mitchell, 1997)
  - 스팸 메일 구분하기—작업 T
  - 훈련 데이터training data—경험 E
  - 정확도accuracy—성능 측정 P

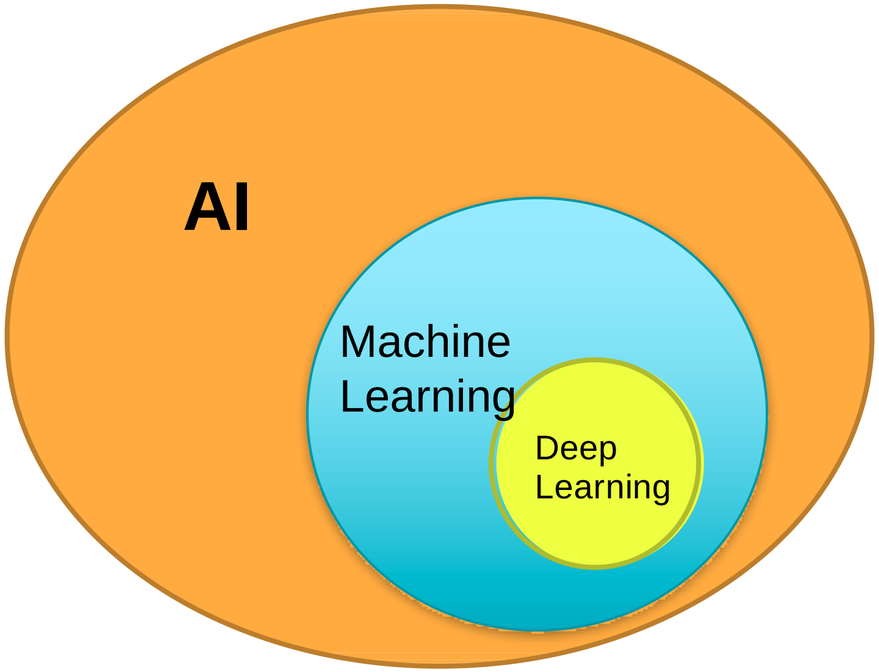
- “머신러닝은 데이터로부터 학습하도록 컴퓨터를 프로그래밍하는 과학(또는 기술)이다.”

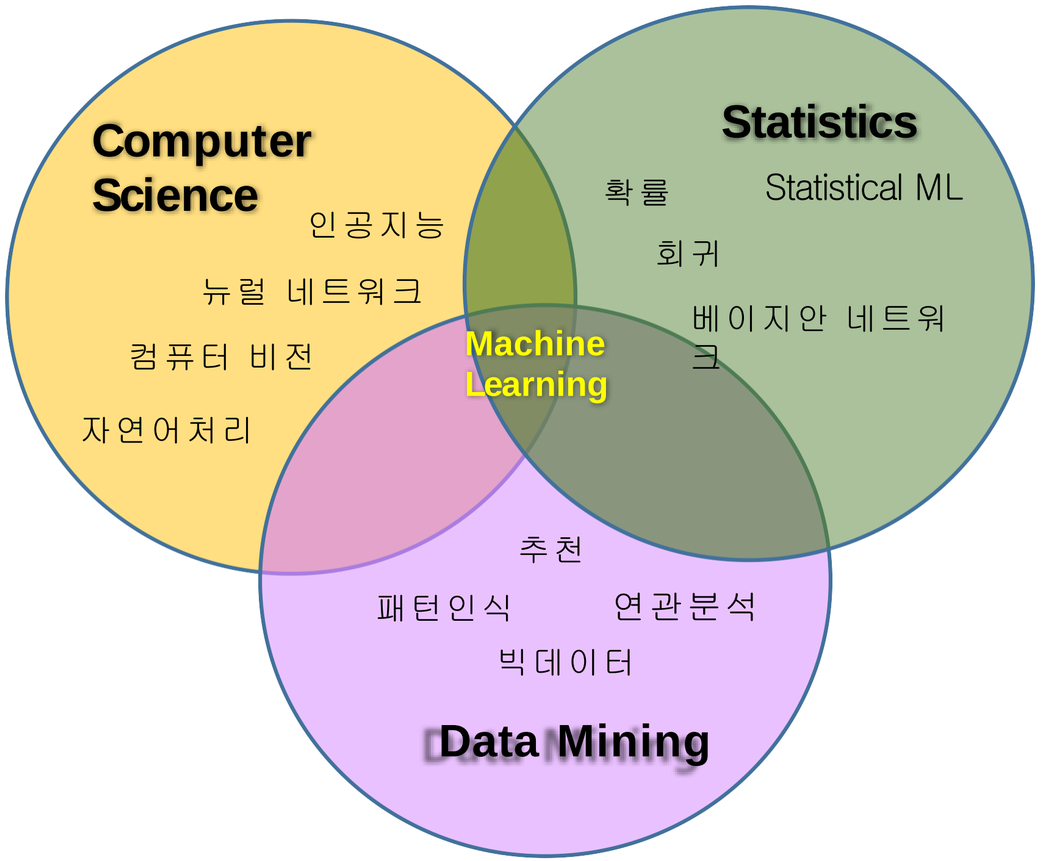
# The Machine Learning Tsunami (교재 서문에서 가져옴)

In 2006, Geoffrey Hinton et al. published a paper1 showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this technique “Deep Learning.” Training a deep neural net was widely considered impossible at the time, and most researchers had abandoned the idea since the 1990s. This paper revived the interest of the scientific community and before long many new papers demonstrated that Deep Learning was not only possible, but capable of mind-blowing achievements that no other Machine Learning (ML) technique could hope to match (with the help of tremendous computing power and great amounts of data). This enthusiasm soon extended to many other areas of Machine Learning.

Fast-forward 10 years and Machine Learning has conquered the industry: it is now at the heart of much of the magic in today’s high-tech products, ranking your web search results, powering your smartphone’s speech recognition, and recommending videos, beating the world champion at the game of Go. Before you know it, it will be driving your car.

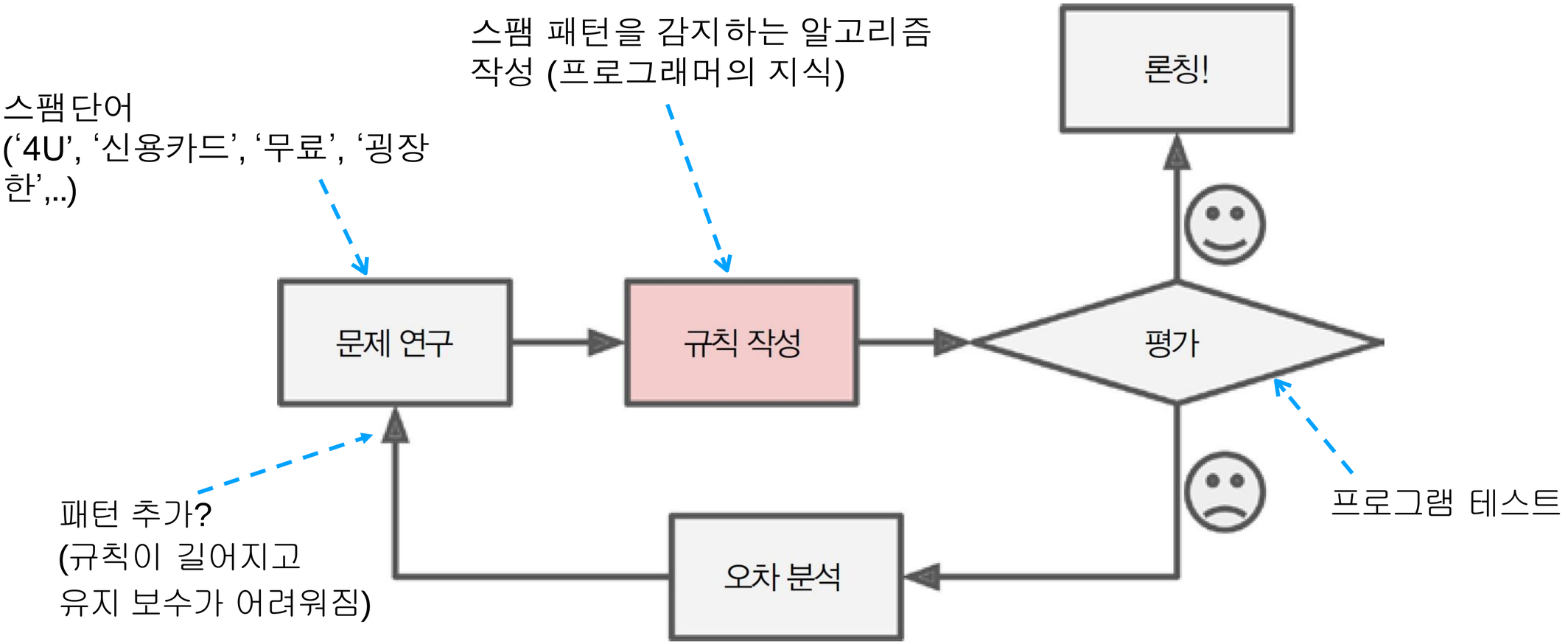
# 인공지능 > 머신러닝 > 딥러닝



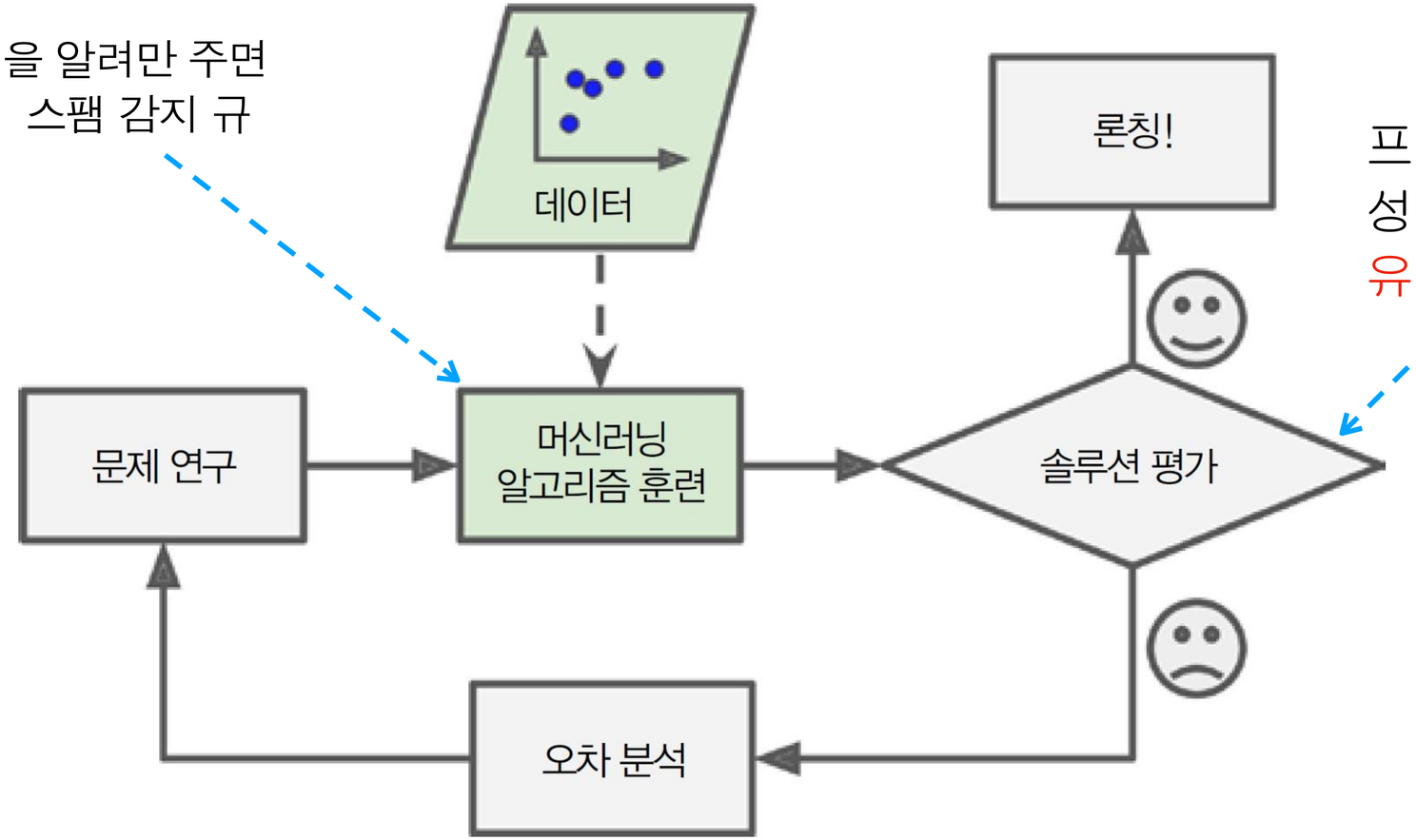
# 왜 머신러닝을 사용하는가

전통적인 접근 방법: 규칙 기반 시스템(rule based system)으로 스팸 필터 프로그램 예



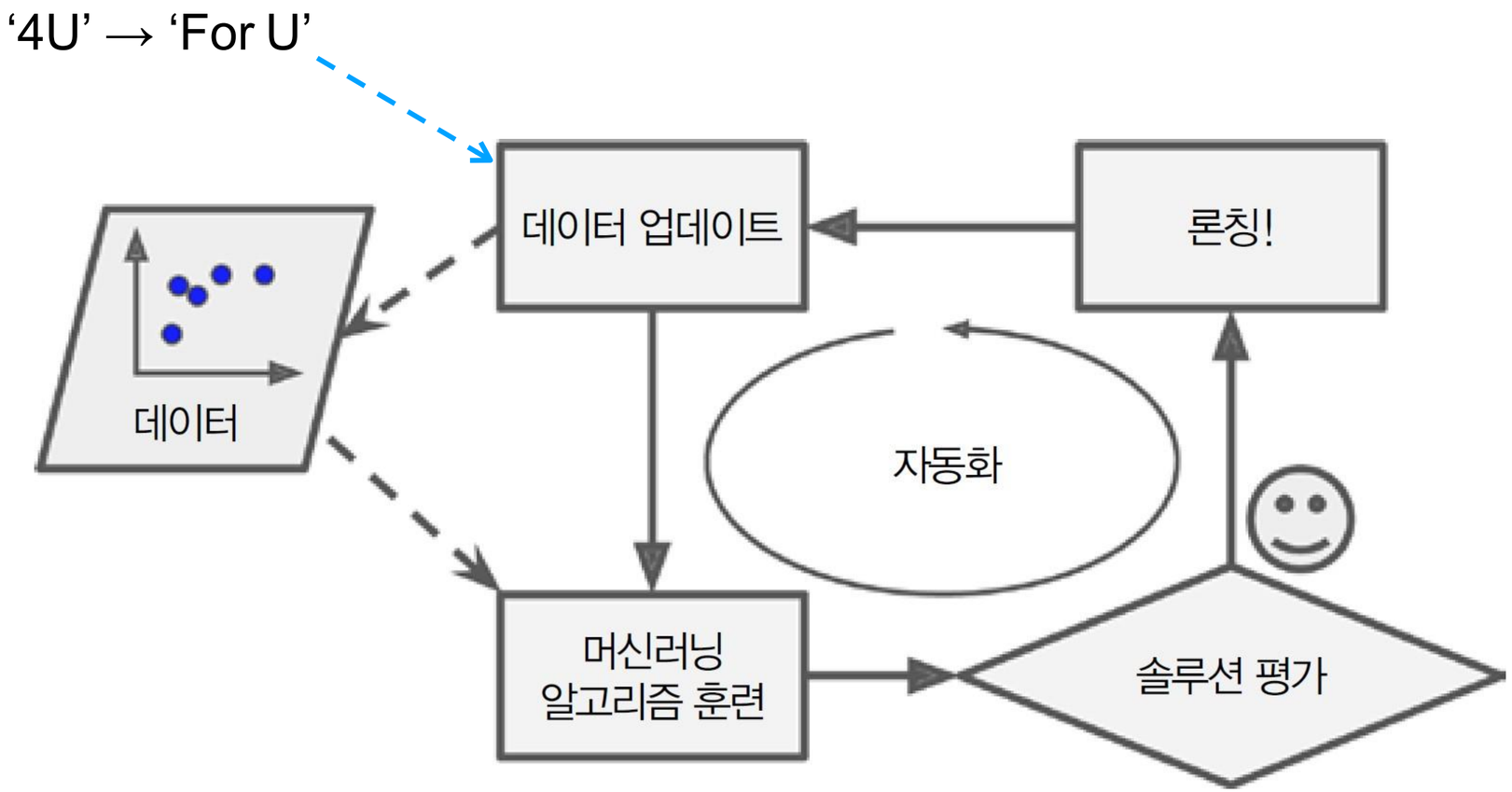
# 머신러닝 접근 방법

사용자가 스팸 메일이라는 것을 알려만 주면 자주 나타나는 패턴 감지하여  스팸 감지 규칙을 자동으로 학습



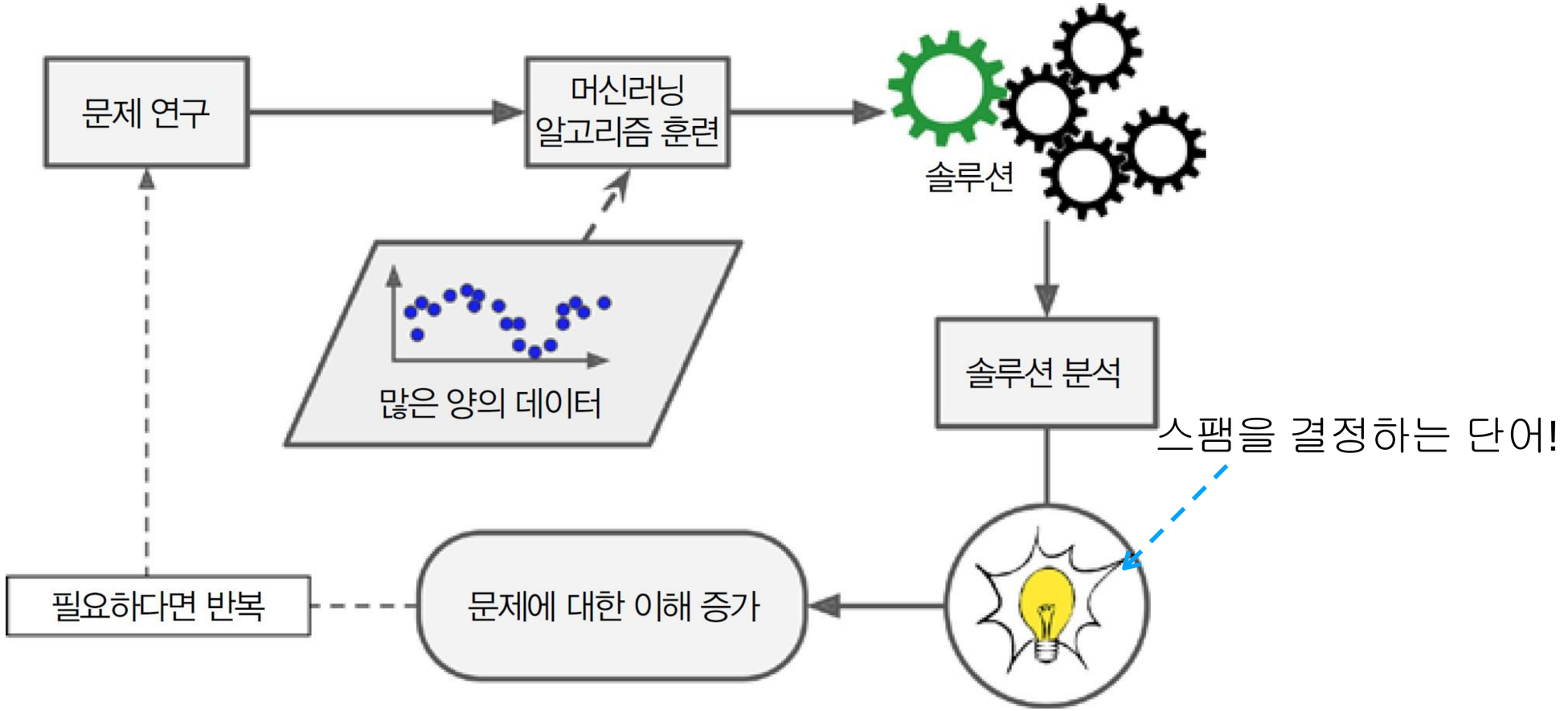

프로그램이 짧고,
성능 우수,
유지 보수 용이

# 자동으로 변화에 적응



# 머신러닝을 통해 새로운 지식을 얻을 수 있음

대용량 데이터를 분석해서 겉으로 보이지 않던 패턴을 발견 :
데이터마이닝(data mining)



# 머신러닝이 유용한 분야

- 많은 수동 조정과 규칙이 필요한 문제
- 전통적인 (지식기반) 방법으로 해결하기 너무 어렵거나 알려진 해가 없는 문제
  - 음성인식
- 변화하는 환경에 적응해야 하는 문제 : 새로운 데이터에 적용할 수 있음
- 복잡한 문제와 대량의 데이터에서 통찰 얻기(데이터 마이닝)

# 머신러닝 시스템의 종류

- 사람의 감독 여부(지도, 비지도, 준지도, 강화학습)
    - Supervised, Unsupervised, Reinforcement
- 실시간으로 점진적 학습인지 여부(온라인 학습, 배치 학습, 점진적 학습)
    - Online, Batch, Incremental
- 훈련 데이터와 단순 비교인지, 패턴을 찾기 위해 예측 모델을 만드는지 (사례 기반, 모델 기반 학습)
    - Instance-based, Model-based

- 이 범주들은 배타적이지 않음(e.g. 스팸 필터-온라인 모델 기반 지도 학습)

# 지도학습(Supervised Learning)

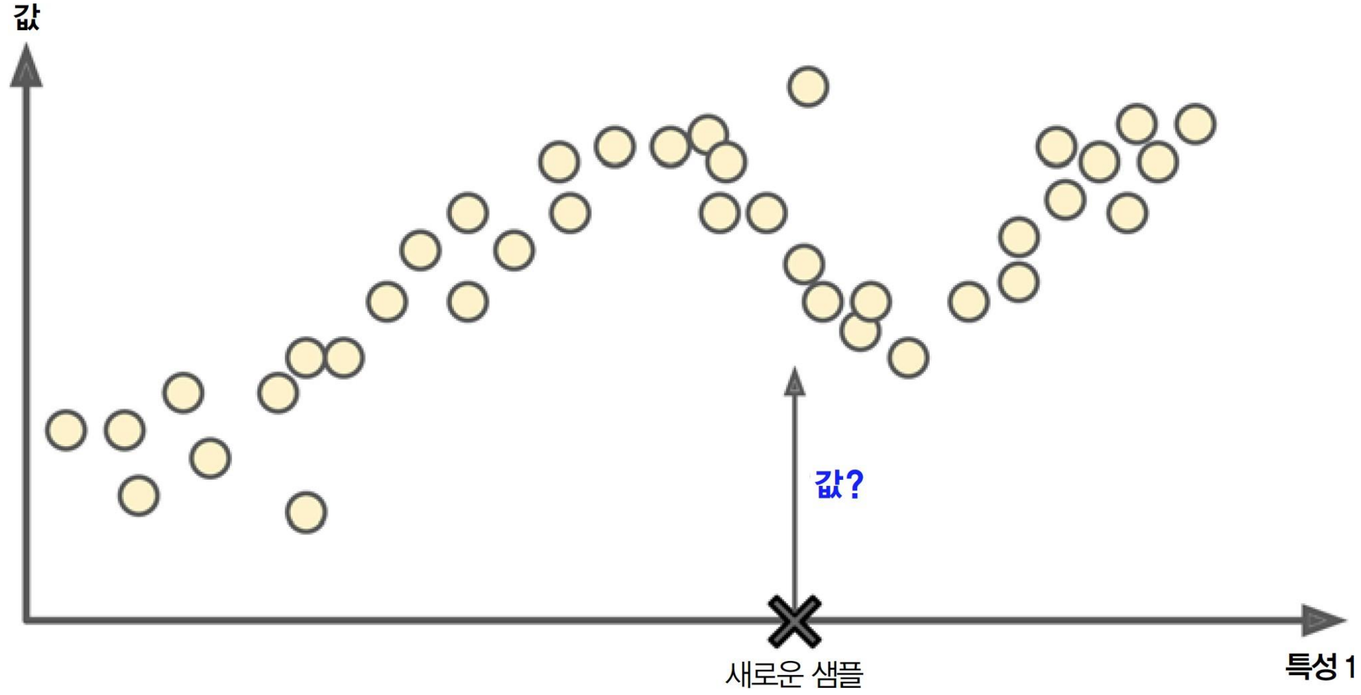
- 학습데이터베이스(훈련세트) = (특징, 레이블) 쌍

  - 레이블[Label. 정답**(ground truth)** 또는 타깃**(target)**이라고도 함] :
    - 스팸 / 스팸아님. ➔ 이진 분류(binary classification)
    - 개 / 고양이 / 새 ➔ 다중 분류(multiclassclassification)
  - 클래스[Class] : 레이블의 범주
    - 스팸 / 스팸아님.
    - 개 / 고양이 / 새
  - 특징[feature. 속성(attribute)]
  - 문서내 문자들 ➔ 스팸 / 스팸아님 : 분류(Classification)
    영상의 픽셀들 ➔ 개 / 고양이 / 새 : 분류(Classification)
    (주행거리, 년식, 브랜드) ➔ 중고차 가격 : 회기(Regression)


교과서에서는 '특성'으로 번역 ➔ '특징'이 일반적인 용어

# 회귀(regression)

- 연속된 숫자를 예측
- 주행거리, 연식, 브랜드 등(특성)을 사용하여 중고차 가격(타깃) 예측

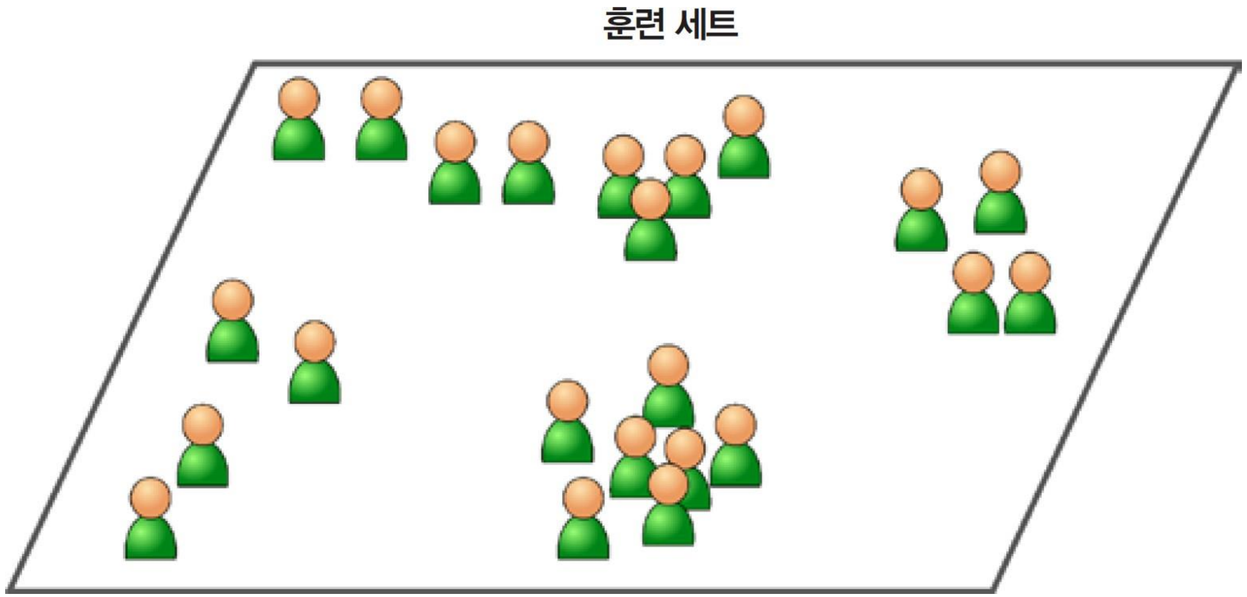


# 대표적인 지도 학습 알고리즘

- k-최근접 이웃(k-Nearest Neighbors)
- 선형 회귀(Linear Regression) - 회귀 알고리즘
- 로지스틱 회귀(Logistic Regression) - 분류 알고리즘
- 서포트 벡터 머신(Support Vector Machine, SVM)
- 결정 트리(Decision Tree), 앙상블 학습(Ensemble Learning)
- 신경망(Neural Network), 딥러닝(Deep Learning) : 지도/비지도/강화

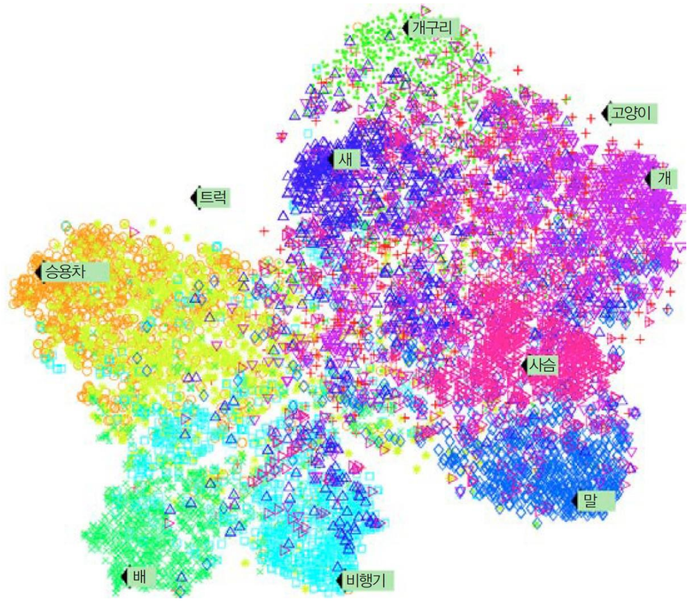
# 비지도 학습(Unsupervised Learning)

- 훈련 데이터에 레이블이 없음
- 블로그 방문자를 그룹으로 묶기 (어떻게 40%가 만화 애호가, 20%는 공상가인지 알 수 있을까?)
- 고객을 자동으로 그룹으로 나누어 각각 다른 홍보자료를 보내기 (캠핑용품, 식료품, 가전제품, ..)



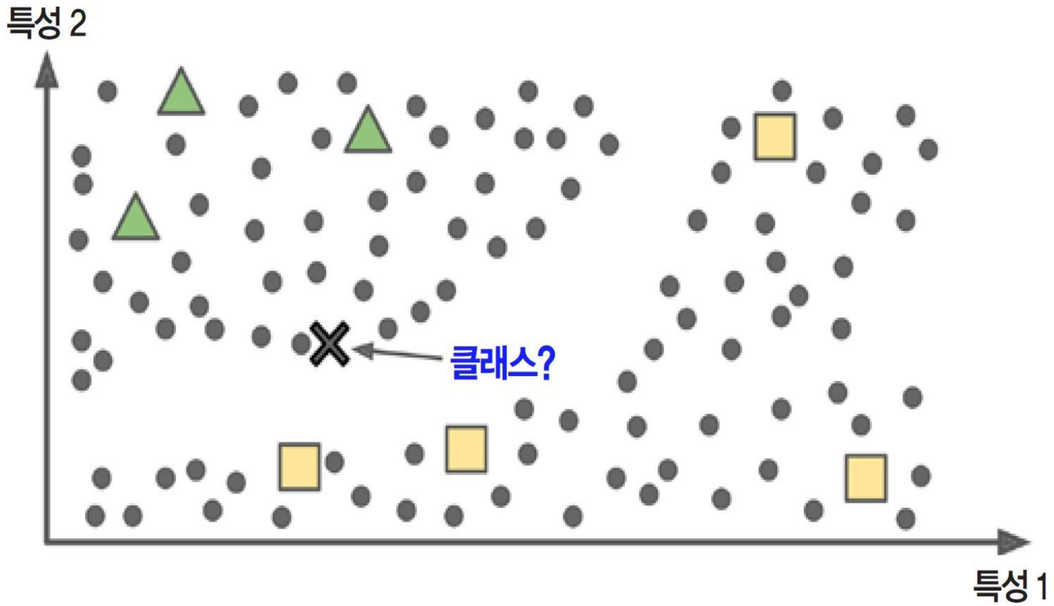
# 대표적인 비지도 학습 알고리즘

- 군집(Clustering)
  - k-평균(k-means)
  - 계층 군집(Hierarchical Clustering): 병합 군집(Agglomerative Clustering)
  - 기댓값 최대화(Expectation Maximization)
- 시각화, 차원 축소-8장
  - 주성분 분석(PCA), 커널 PCA, NMF
  - 지역 선형 임베딩(LLE)
  - t-SNE
- 연관 규칙(Association Rule): 데이터 마이닝
  - 어프라이어리(Apriori), 이클렛(Eclat)

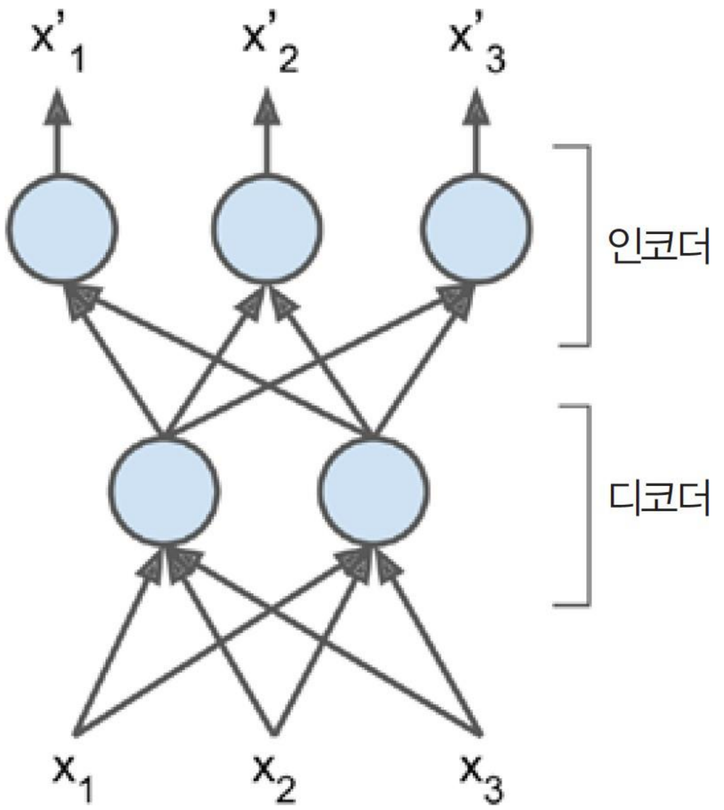
# 준지도 학습(Semi-supervised Learning)

- 대부분 데이터에는 레이블이 없고 일부 데이터에만 레이블이 있음
- 군집을 통해 데이터를 분할하고 소수 레이블을 이용해 전체 그룹을 인식.
- Google Photos, Facebook 사진 태그
- 제한된 볼츠만 머신(RBM)으로 구성된 심층 신뢰 신경망(DBN)



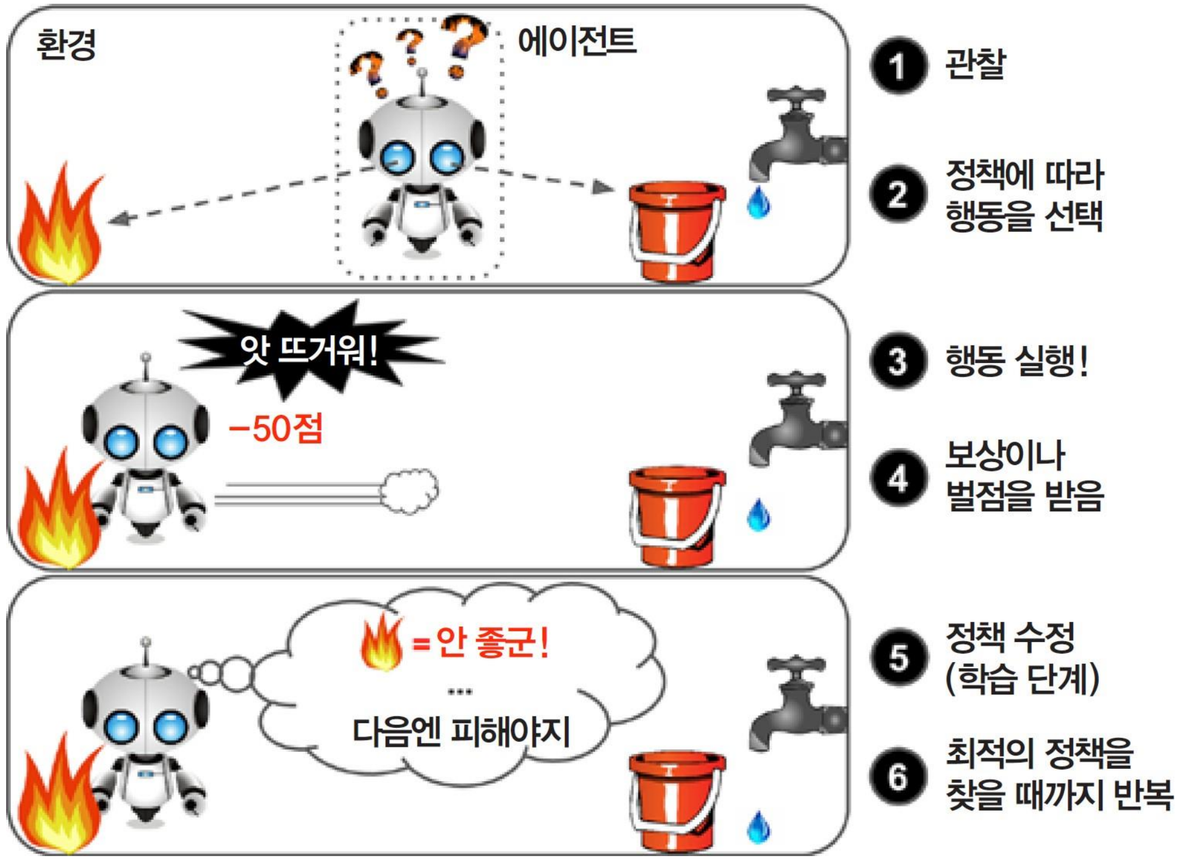
# 자기지도 학습(Self-supervised Learning)

- 오토인코더(Autoencoder)-15장
- 입력과 레이블이 동일함.
- 일반적인 레이블이 없기 때문에 비지도 학습, 자기자신이 타깃이기 때문에 자기지도 학습, 어쨋든 타깃이 있으므로 지도 학습으로도 부름.



# 강화학습(Reinforcement Learning)

- 행동 결과에 따른 보상(혹은 벌점)을 레이블로 줌 : 바둑 경우 이겼다 혹은 졌다
- 주어진 환경(environment)에서 에이전트(agent)가 최대의 보상(reward)을 얻기 위해 최상의 정책(policy)을 학습.
- 딥마인드의 알파고(Alpha Go), 아타리(Atari) 게임

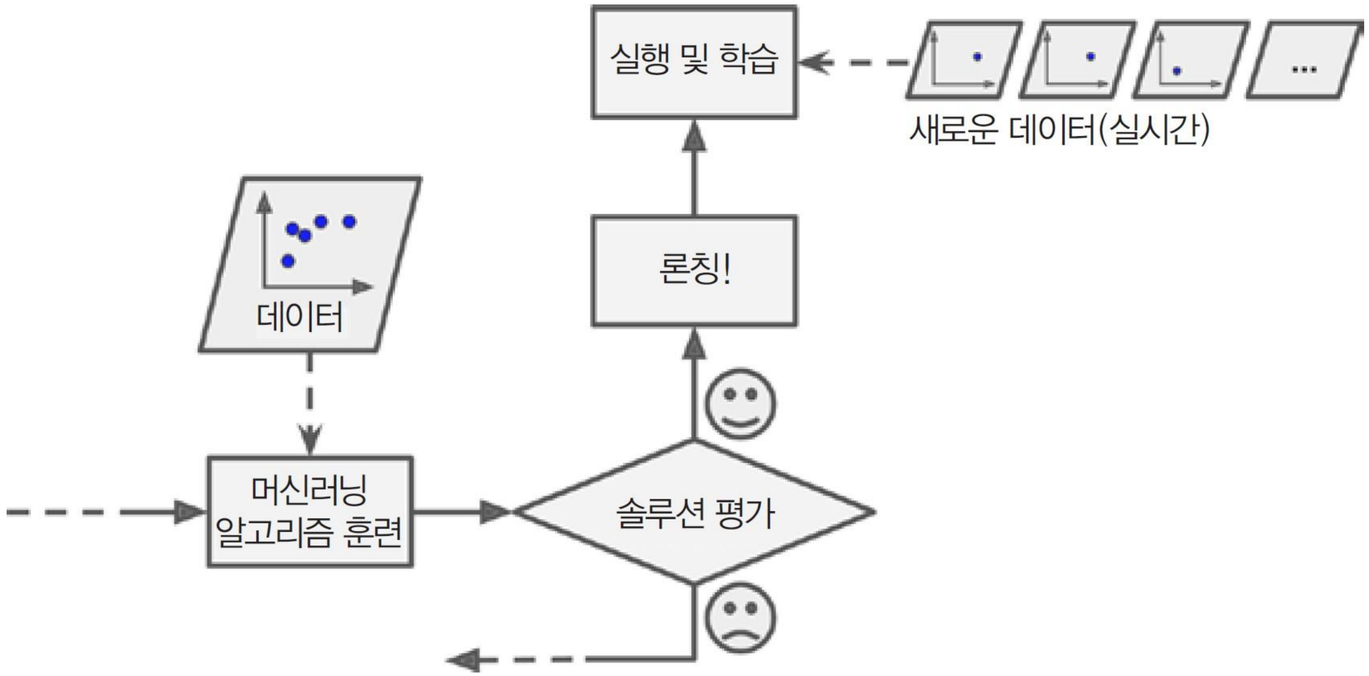


# 배치(batch) 학습

- 학습 데이터를 모두 사용하여 훈련시키는 오프라인(offline) 학습
- 새 데이터가 추가되면 이전 학습데이터와 합쳐 새로 학습함.
- 일반적으로 시간과 자원이 많이 소모됨
- 훈련과 모델 론칭을 자동화할 수도 있음
  - 예) 새 데이터를 추가해서 24시간, 혹은 1주일마다 학습

# 온라인(online) 학습

- 샘플 한 개 또는 미니배치(mini-batch)라 부르는 작은 묶음 단위로 훈련
  ➔ 추가학습(incremental learning)이 더 올바른 표현
- 학습 단계가 빠르고 데이터가 준비되는 대로 즉시 학습할 수 있음(주식 가격?)
- 사용한 샘플을 버릴 수도 있고 보관할 수도 있음

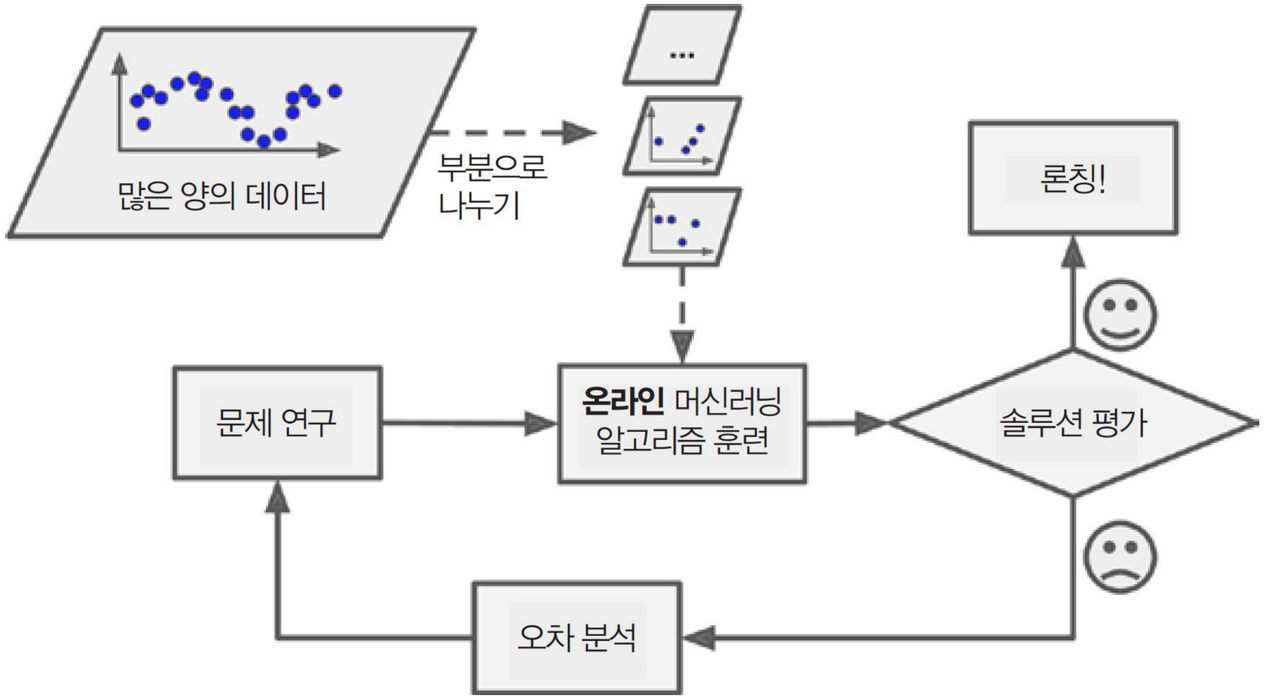


# 온라인 학습의 주의 사항

- 학습률(learning rate)로 데이터에 얼마나 빠르게 적응할지 제어
- 나쁜 데이터가 모델 학습에 주입되면 성능이 조금씩 감소됨
- 시스템 모니터링이 필요하고 성능 감소가 감지되면 학습을 중지하고 이전 버전으로 되돌리거나 비정상적인 데이터를 찾음

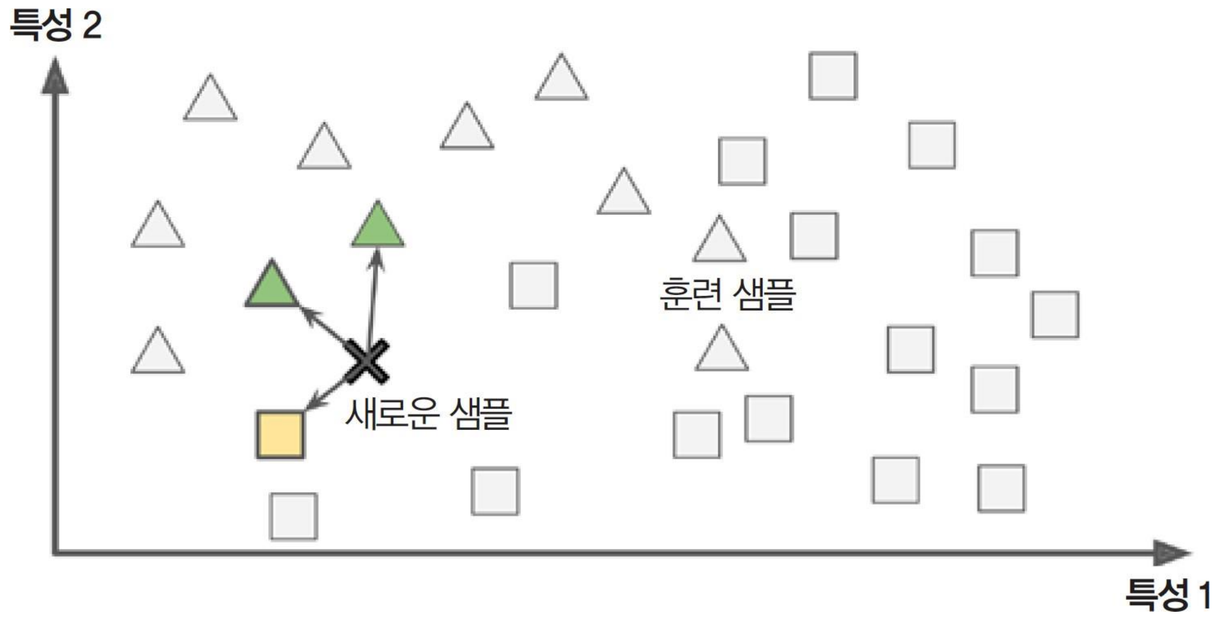
# 외부 메모리(ooc) 학습

- 데이터를  부분으로 나누어 오프라인에서 학습
- Out-of-core learning
- 딥러닝 처럼 데이터가 많은 경우 사용 : 미니배치학습(Mini Batch Learning)



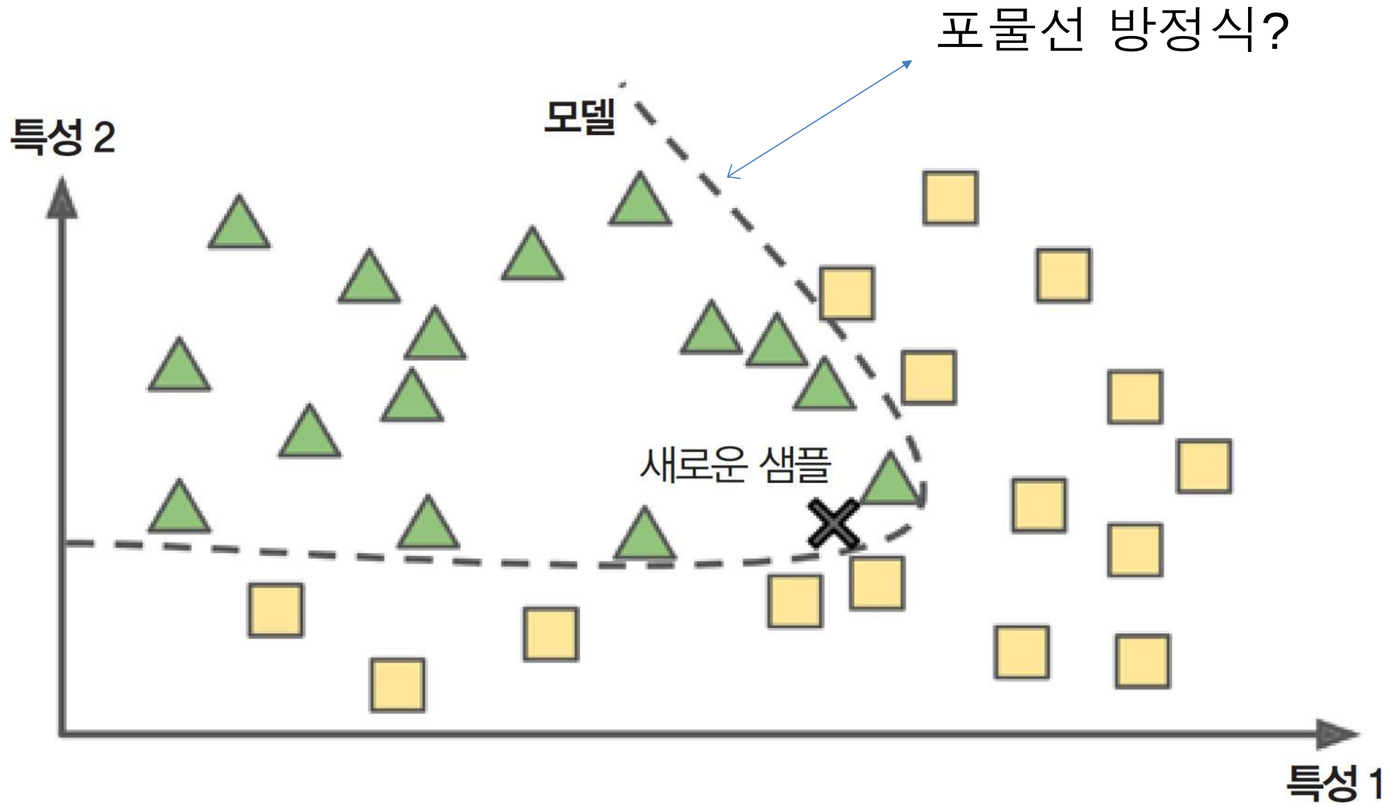
# 사례 기반 학습 (instance-based learning)

- 사례 기반(instance-based) : 학습데이터를 그대로 사용
  모델 기반(model-based) : 학습데이터를 대표하는 모델을 만들어 그 모델을 학습함
- k-최근접 이웃(k-Nearest Neighbors) : 저장된 훈련 데이터에서 가장 가까운 샘플을 찾음. (학습이 필요 없거나 간단함) ➔ 사례기반 학습



# 모델 기반 학습

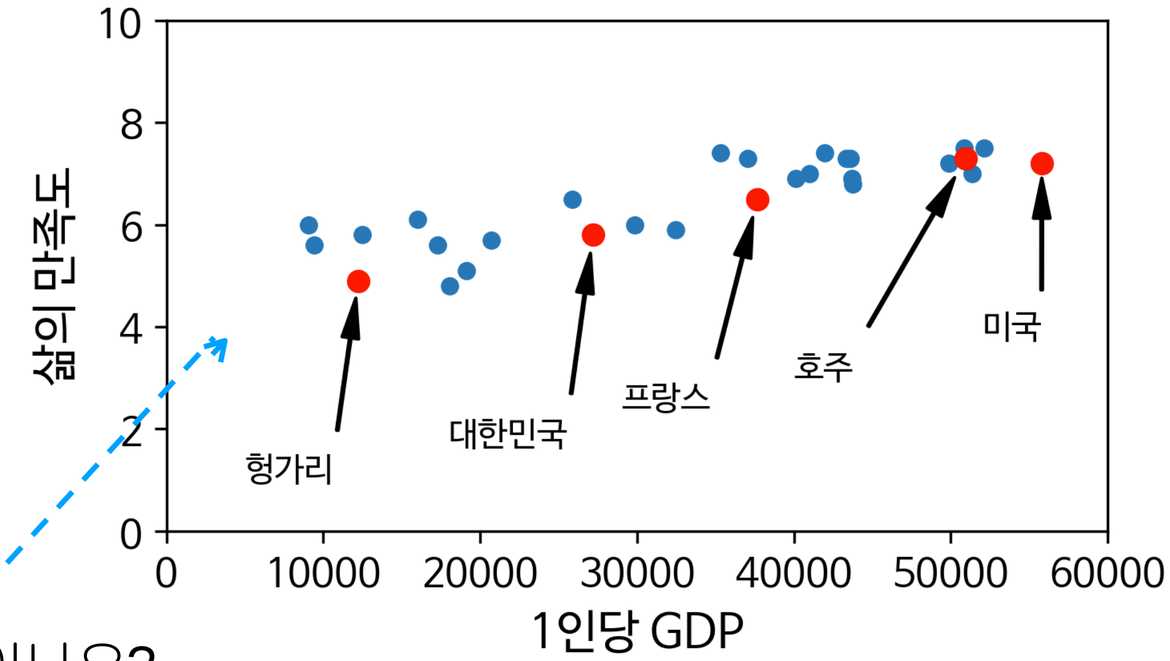
- 모델 기반(model-based) : 학습데이터를 대표하는 모델을 만들어 그 모델을 학습함
- 대부분 모델은 파라메터(parameter)를 가지고 있음 ➔ 학습은 최적의 파라메터값을 결정하는 것
- 거의 대부분의 머신러닝은 모델 기반 학습



# 1인당 GDP에 대한 삶의 만족도

- OECD 웹사이트에서 '더 나은 삶의 지표'와 IMF 웹사이트에서 '1인당 GDP' 데이터를 다운로드하여 사용 : 교재에서 제공하는 code 참조 *https://github.com/ageron/handson-ml*

| 국가 | 1인당 GDP(US달러) | 삶의 만족도 |
|---|---|---|
| 헝가리 | 12,240 | 4.9 |
| 대한민국 | 27,195 | 5.8 |
| 프랑스 | 37,675 | 6.5 |
| 호주 | 50,962 | 7.3 |
| 미국 | 55,805 | 7.2 |

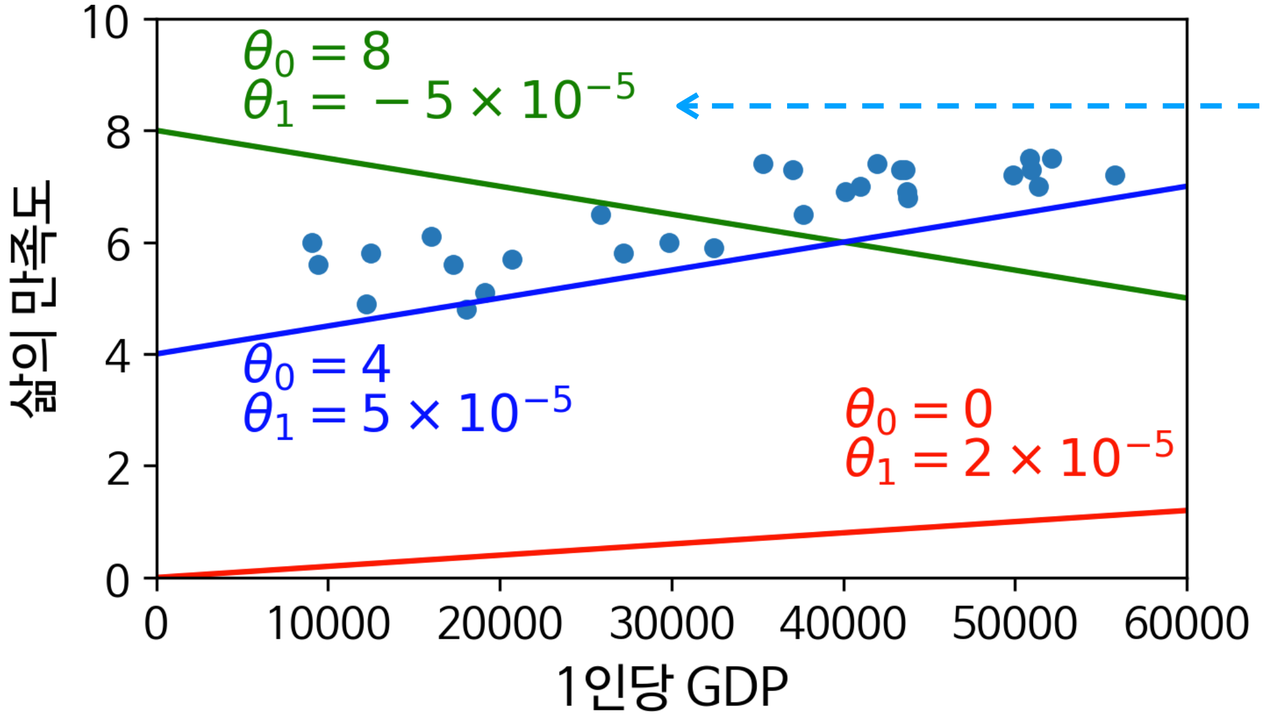



어떤 경향이 보이나요?

# 모델 선택

- 1인당 GDP의 선형 함수(linear function)로 삶의 만족도를 모델링 : 선형 회귀

삶의 만족도 = $\theta_0$ + $\theta_1$ × 1인당 GDP




모델 파라미터(model parameter)

어떤 모델이 가장 좋은가?

# 비용 함수(Cost Function)

- 좋은 모델을 고르기 위해 얼마나 나쁜지를 측정.
- 목적 함수(Object function) > 비용 함수 > 손실 함수(Loss function)
- 선형 회귀에서의 비용함수 : 예측과 타깃 사이의 거리의 합
  ➔ 비용이 클 수로 나쁜 모델
- 주어진 데이터에서 비용 함수의 값이 가장 작아지는 모델 파라미터( $\theta_0, \theta_1$ )를 찾는 과정을 훈련(training) 또는 학습(learning)이라고 함
- 학습된 모델을 이용해서 입력 샘플에 대한 출력값을 계산하는 과정은 예측(predict) 혹은 시험(test)이라고 함

# 훈련(학습)

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model

# 데이터 적재
oecd_bli = pd.read_csv(datapath + "oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv(datapath + "gdp_per_capita.csv",thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")

# 데이터 준비
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# 데이터 시각화
ax = country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
ax.set(xlabel="삶의 만족도", ylabel="1인당 GDP")
plt.show()

# 선형 모델 선택
model = sklearn.linear_model.LinearRegression()

# 모델 훈련
model.fit(X, y)
```

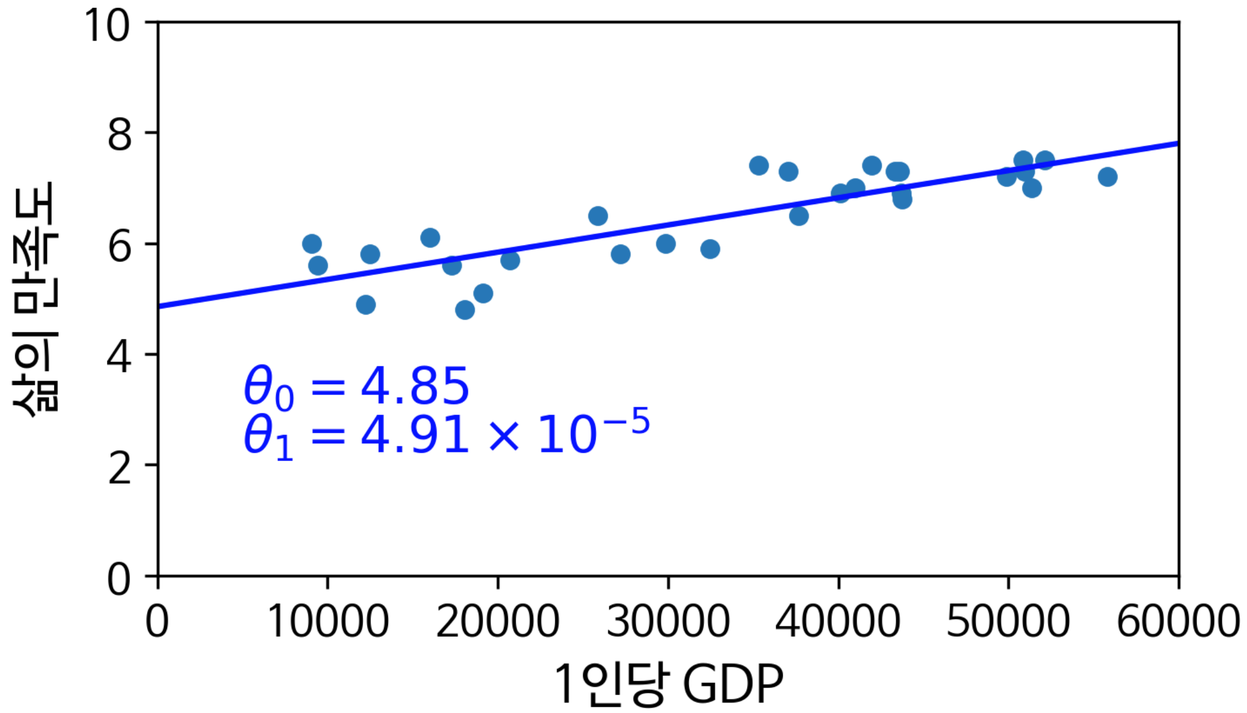
```
# oecd_bli, gdp_per_capit 를 나라이름으로 병합한 후, "GDP per capita " 와 'Life satisfaction’ 등 두개 컬럼만 남김
def prepare_country_stats(oecd_bli, gdp_per_capita):
    oecd_bli = oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli = oecd_bli.pivot(index="Country", columns="Indicator", values="Value")
    gdp_per_capita.rename(columns={"2015": "GDP per capita"}, inplace=True)
    gdp_per_capita.set_index("Country", inplace=True)
    full_country_stats = pd.merge(left=oecd_bli, right=gdp_per_capita,
                                  left_index=True, right_index=True)
    full_country_stats.sort_values(by="GDP per capita", inplace=True)
    remove_indices = [0, 1, 6, 8, 33, 34, 35]
    keep_indices = list(set(range(36)) - set(remove_indices))
    return full_country_stats[["GDP per capita", 'Life satisfaction']].iloc[keep_indices]


# 파일 읽어들일 폴더이름 설정
import os
datapath = os.path.join("datasets", "lifesat", "")
```

# 학습결과 : 모델 파라미터

- model.intercept_, model.coef_

  삶의 만족도 = 4.85 + 4.91 × $10^{-5}$ × 1인당 **GDP**



# 예측 : 키프로스의 삶의 만족도?

- 키프로스의 1인당 GDP는 $22,587
  삶의 만족도 = $4.85 + 4.91 \times 10^{-5} \times 22587 = 5.96$

```
# 키프로스에 대한 예측
X_new = [[22587]]  # 키프로스 1인당 GDP
print(model.predict(X_new)) # 결과 [[ 5.96242338]]
```

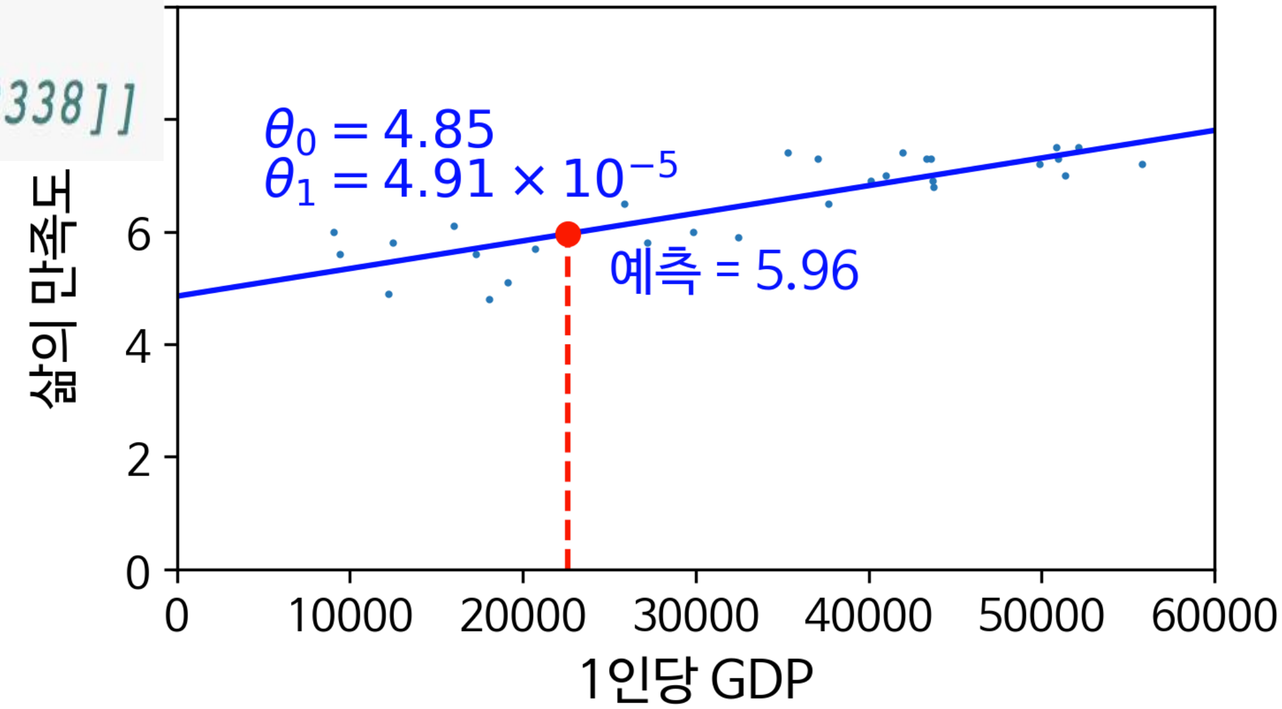
- Scikit-Learn: fit(X, y) → predict(X)

특징 (행렬)

레이블 (벡터)

학습 데이터베이스

**시험샘플 (예측하려고 하는 데이터)**



# k-최근접 이웃을 사용하면?

- k=3일 때, 키프로스와 가장 가까운 슬로베니아, 포르투갈, 스페인의 삶의 만족도를 평균
- fit(X, y) → predict(X)

```
# 선형 회귀 모델을 k-최근접 이웃 회귀 모델로 교체할 경우
knn = sklearn.neighbors.KNeighborsRegressor(n_neighbors=3)

# 모델 훈련
knn.fit(X, y)

# 키프로스에 대한 예측
print(knn.predict(X_new)) # 결과 [[ 5.76666667]]
```

# 무언가 잘 못 되었다면?

- 특징 : 더 많은 특징(고용률, 건강, 대기오염 등)을 사용
- 데이터 : 좋은 훈련 데이터를 더 많이 모음
- 모델 : 더 강력한 모델(다항 회귀 모델 등)을 선택

# 머신러닝 작업 요약

- 데이터를 분석
- 모델을 선택
- 훈련 데이터로 모델을 훈련. (비용 함수를 최소화하는 모델 파라미터를 찾음)
- 새로운 데이터에 대한 예측(추론)을 함. (좋은 일반화를 기대)
- 그리고 반복!

추론(inference) : 특징과 결과와의 관계에 중점,
예측(prediction) : 결과값에 중점 ➔ 이 두 단어는 같은 의미로 사용

# 머신러닝의 도전 과제

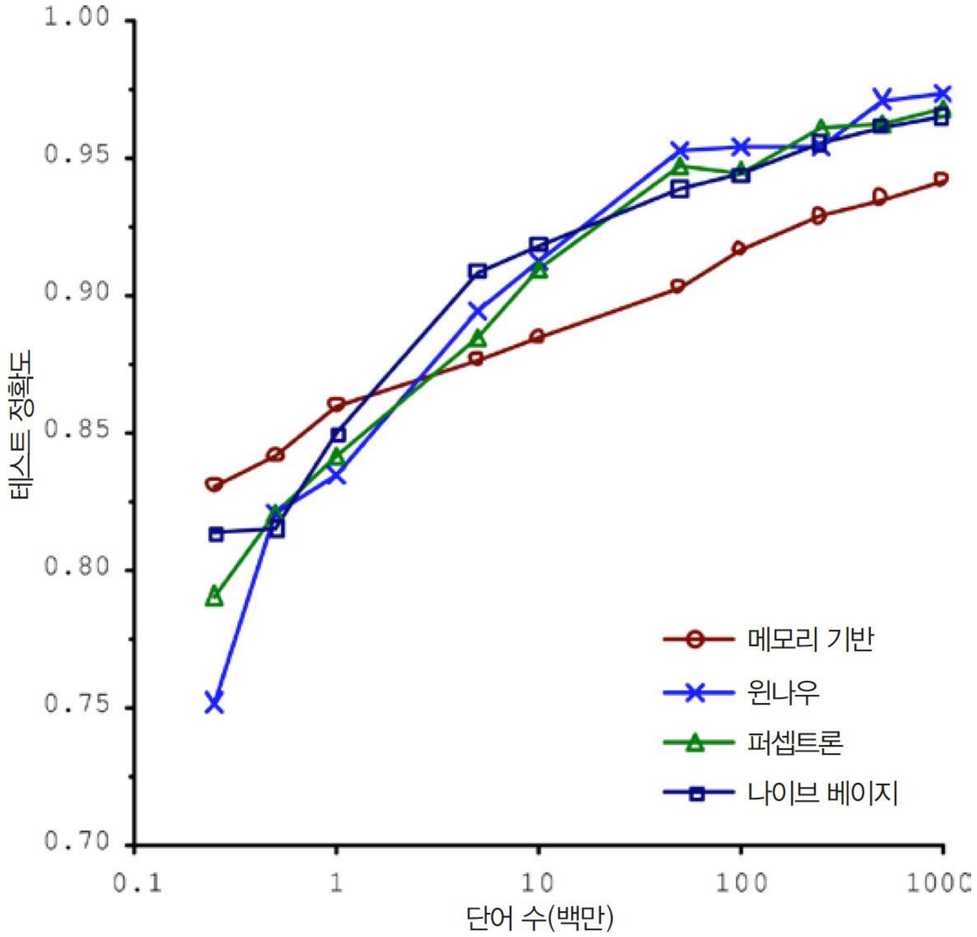
- 나쁜 데이터
  - 충분하지 않은 훈련 데이터
  - 대표성 없는 훈련 데이터
  - 낮은 품질의 데이터
  - 관련 없는 특성
- 나쁜 알고리즘
  - 과대적합
  - 과소적합

# 충분하지 않은 훈련 데이터

- 어린 아이는 '사과' 샘플 몇 개를 보고도 모든 종류의 사과를 쉽게 일반화 함

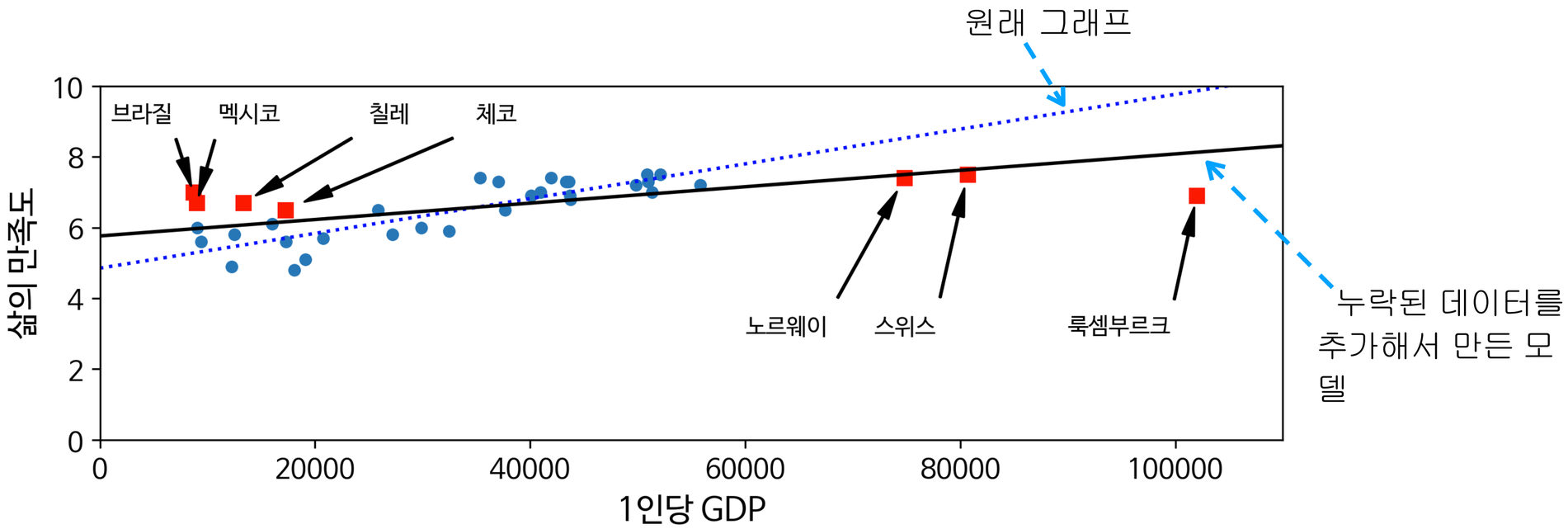
- 머신러닝은 데이터로부터 일반화 규칙 학습
  ➔ 많은 데이터가 필요함

- 2001년 MS 연구자들은 충분한 데이터가 주어지면 알고리즘들이 복잡한 자연어 처리 문제를 거의 비슷하게 잘 처리한다는 것을 보임 : 데이터 중요
  (실제로는 데이터가 제한적이므로 알고리즘도 중요함)



# 대표성 없는 훈련 데이터

- 삶의 만족도 모델에 누락된 데이터를 추가하면 그래프가 달라짐




- 샘플링 잡음(sampling noise): 샘플이 작거나 이상치가 포함된 경우
- 샘플링 편향(sampling bias): 표본 추출 방법이 잘 못 된 경우

# 가장 유명한 샘플링 편향 사례

- 1936년 랜던과 루즈벨트의 대통령 선거에서 Literary Digest 사의 여론조사(랜던 57% 예측)


- 실제로는 루즈벨트가 60.8% 득표로 당선됨
  ➔ 샘플링 편향으로 잘못된 결과

1) 전화번호부, 구독자 명부, 클럽 회원 명부를 사용
   ➔ 랜던에 투표할 가능성이 높은 부유층

2) 우편물 수신자 중 25% 미만이 응답
   ➔ 정치에 관심없는 사람, 조사기관(Literary Digest)을 싫어하는 사람이 제외됨 (비응답 편향)

# 낮은 품질의 데이터

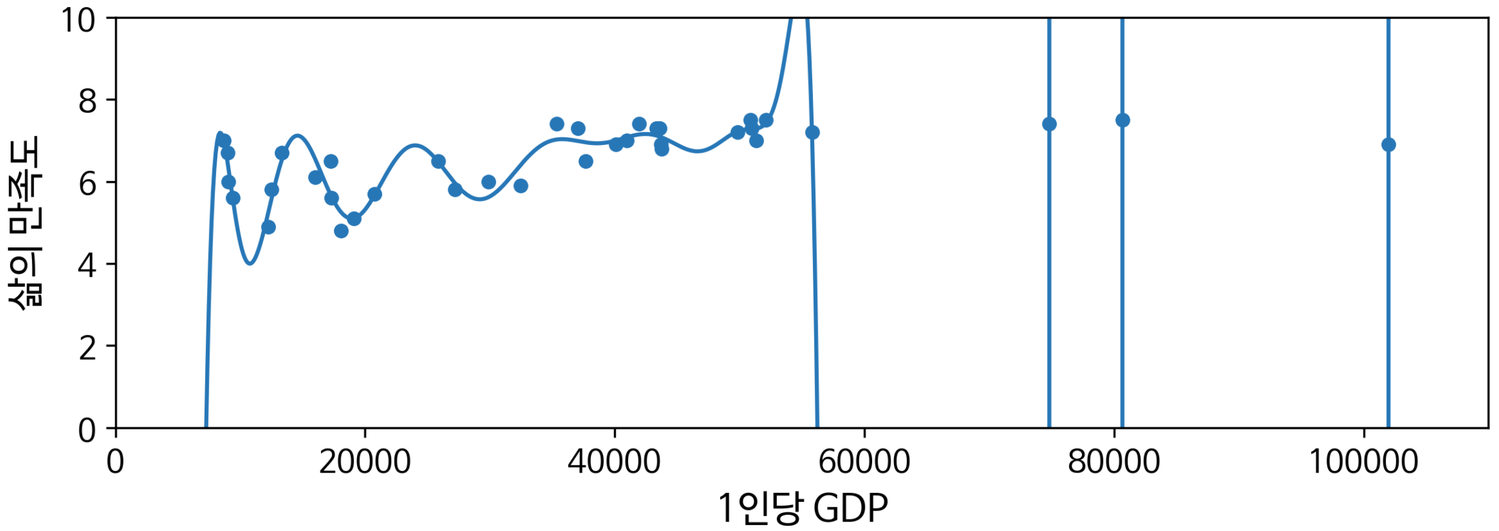
- 일부 샘플이 이상치일 경우 무시하거나 수동으로 고침.
- 일부 샘플에서 특성이 몇 개 빠져 있다면, 특성 전체를 무시할지, 샘플을 무시할지, 빠진 값을 채울지, 이 특성이 넣은 것과 뺀 것을 따로 훈련할지 정해야 함.

# 특징 공학

- 주어진 문제와 관련이 높은 특징을 찾음 : 특징 공학(feature engineering)
  ➔ 특징이 많다고 잘 분류하는 것이 아님. “좋은 특징”을 사용하는 것이 중요함
  - 특징 선택: 가지고 있는 특징 중에서 가중 유용한 특징을 선택.
  - 특징 추출: 특징을 결합하여 더 유용한 특징을 만듬(차원 축소, 다항 회귀 등)
  - 새로운 데이터로부터 새로운 특징을 만듬

# 과대적합(overfitting)

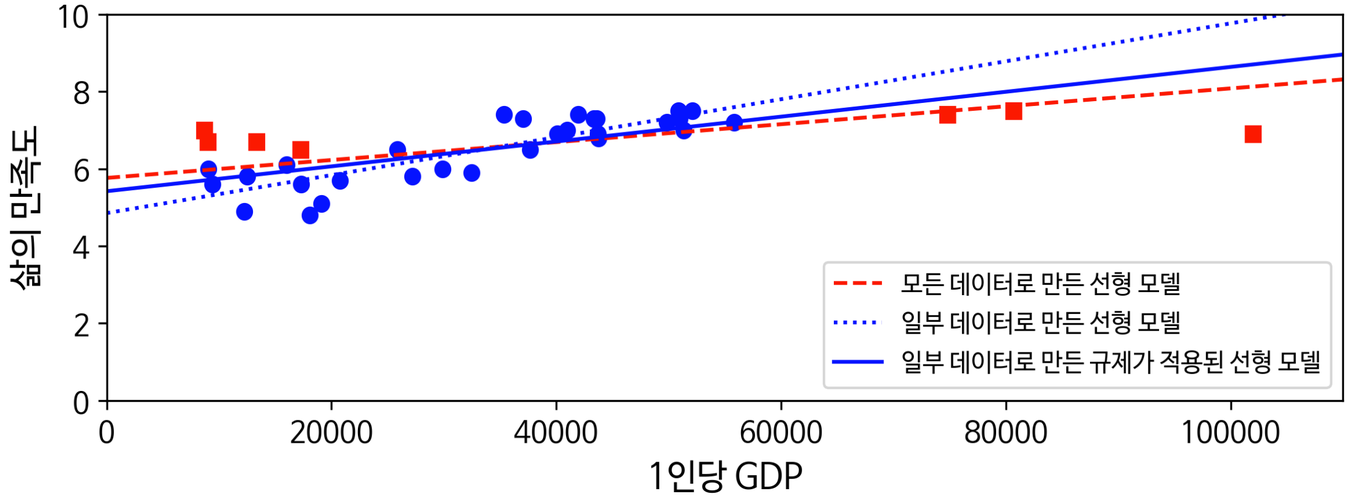
- 해외 여행에서 택시 운전사에게 속았다면 그 나라의 모든 택시를 의심하게 됨
  ➔ 일반화의 오류 (=과대적합)
- 학습데이터(==훈련데이터)에만 잘 들어 맞음. 시험데이터에는 잘 안 맞음
- 이유 :
  - 너무 복잡한 모델을 사용
  - 잡음이 많거나 너무 적은 수의 샘플(=학습샘플=학습데이터)



# 과대적합을 피하려면

- 데이터 측면 :
  - 학습데이터 수 늘림
  - 학습데이터의 잡음 감소 : 오류 데이터 수정, 이상치 제거
- 모델 측면 :
  - 모델 파라미터 개수가 적은 모델(복잡도가 낮은 모델)을 선택
  - 특징 수를 줄임 : 특징공학 방법 사용
  - 모델에 regularization(**정규화**, 혹은 '규제'로 번역)를 추가

- Regularization : 모델 파라미터의 값에 제한 조건을 줌.
  - 크기의 절대값의 합을 제한 : L1
  - 크기의 제곱합을 제한 : L2



# 과소적합(underfitting)

- 과대적합의 반대. 모델이 너무 단순해서 적절한 패턴을 학습하지 못함

- 해결 방법
  - 모델 파라미터가 더 많은 (복잡도가 높은) 모델을 선택
  - 더 좋은 특징 사용 : 특징 공학
  - 모델의 Regularization 축소

Regulariazation을 얼마나 할 것인가는 사용자가 결정하는 파라메터임.
이는 학습에 의해 결정되는 모델 파라메터와 다름.
학습되지 않고 결정되는 파라메터를 하이퍼파라메터(Hyperparameter)라고 함

# 한걸음 물러서서

- 머신러닝은 명시적인 규칙을 코딩하지 않고 데이터에서 학습함

- 지도 학습 / 비지도 학습,
  배치 학습 / 온라인 학습,
  사례 기반 학습 / 모델 기반 학습

- 학습 : 학습 데이터를 잘 표현하는 모델 파라미터를 찾음

- 데이터 : 개수가 너무 작거나, 대표성이 없거나, 잡음이 많거나, 관련 없는 특징이 많으면  올바른 모델을 학습하지 못함

- 모델이 너무 복잡하거나(과대적합), 단순하지(과소적합) 않아야 함.

# 테스트 세트와 검증 세트

- 훈련된 모델을 검증하기 위해 따로 떼어 놓은(훈련에 사용하지 않은) 테스트 세트를 사용해야 함

- 훈련 세트(training set): 80%, 테스트 세트(test set): 20%

- 하이퍼파라메터을 조정하기 위해 검증세트(validation set)가 사용되기도 함
  (테스트 세트를 사용하여 하이퍼파라미터를 조정하면 모델이 테스트 세트에 과대적합됨)
  예) 훈련세트 60%, 검증세트 20%, 테스트세트 20%

    - 훈련 세트와 검증 세트를 교번하여 여러번 검증 점수를 계산(교차 검증(cross-validation)).

# 공짜 점심 없음(no free lunch)

- 어떤 가정도 없다면 특정 모델이 뛰어나다고 판단할 근거가 없다는 이론 : 데이비드 월퍼트(1996)

- 주어진 데이터셋에 선형 모델이 잘 맞을지 신경망이 잘 맞을지 경험하기 전에 알 수 없음 : 결국 모두 시도해 보아야 함

- 일반적인 모델 선정 가이드 :
  - 정형화된 데이터 :  트리기반 앙상블
  - 지각에 관련된 데이터(이미지, 텍스트, 사운드) : 신경망