



# [스파르타코딩클럽] 가장 쉽게 배우는 머신러닝 - 1주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

## ▼ PDF 파일

### [수업 목표]

1. 머신러닝의 기초 개념을 알아본다.
2. 선형회귀에 대해 배운다.
3. Colab과 Kaggle을 이용해 직접 실습해본다!

### [목차]

- 01. 1주차 오늘 배울 것
- 02. 필수 계정 가입하기
- 03. 머신러닝이란?
- 04. 선형 회귀 (Linear Regression)
- 05. 경사 하강법 (Gradient descent method)
- 06. 데이터셋 분할
- 07. 실습 환경 소개 (Colab)
- 08. 간단한 선형회귀 실습
- 09. 캐글 선형회귀 실습
- 10. 1주차 끝 & 숙제 설명
- 11. 1주차 숙제 답안 코드



모든 토글을 열고 닫는 단축키

Windows : **ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

## 01. 1주차 오늘 배울 것

### ▼ 1) 머신러닝

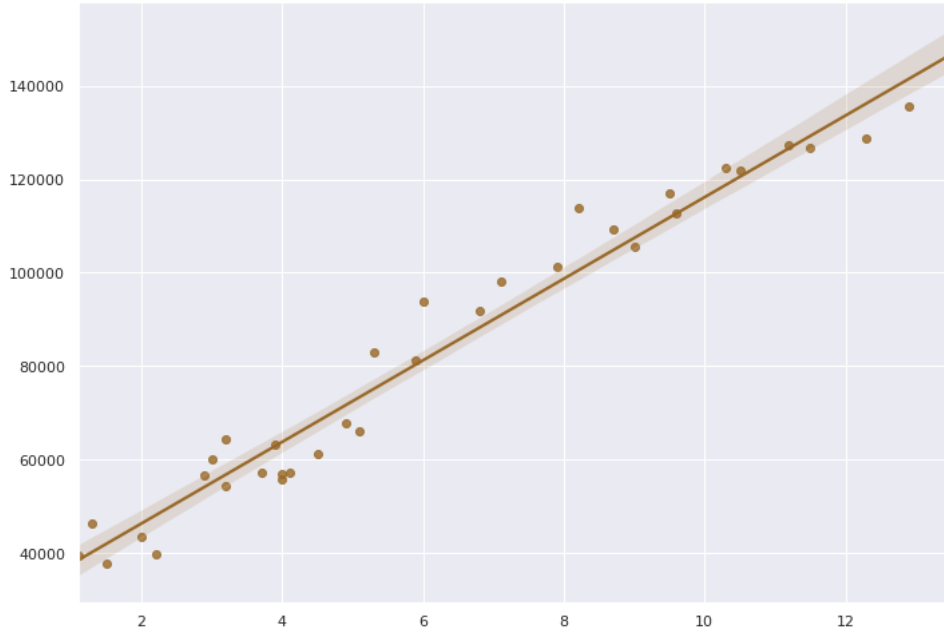


가장 쉽게 배우는 머신러닝에 오신 여러분을 환영합니다!  
기계를 학습시킨다고 하는데, 기계가 뭘 배운다는 건지, 어떻게 가르치는지 궁금하셨죠?  
이번주에 머신러닝에 대해 알아보고 직접 코드도 실행하며 익혀봅시다 😊

### ▼ 2) 선형 회귀



컴퓨터가 풀 수 있는 문제 중에 가장 간단한 것이 바로 두 데이터 간의 직선 관계를 찾아 내서 x값이 주어졌을 때 y값을 예측하는 것인데요! 이것을 **선형회귀**라고 합니다 😊  
머신러닝에서 이런 문제를 어떻게 푸는지, 어떤 개념들이 필요한지 알아보아요~



### ▼ 3) 강의에 들어가기에 앞서

#### ▼ 1) 영어로 된 용어를 사용하길 권장하는 이유

1. 구글, Stackoverflow 등의 사이트에서 영어를 많이 씀
2. 의사소통시 영어로 소통해야 의사소통 오류가 적음
3. 외국인 엔지니어와의 의사소통
4. 외국계 기업 취업
5. 심하게 아는 척 가능 😎

#### ▼ 2) (질문보다) 구글링의 중요성

1. 실무에서 실제로 구글링이 차지하는 비율이 90% 이상
2. 검색 능력이 곧 업무 성과에 비례하는 경우가 많음
3. 새로운 분야를 개척할 수록 질문할 사람(사수)이 없는 경우가 많음

## 02. 필수 계정 가입하기



수업 실습에서 사용하는 **Google Colab**과 **Kaggle**을 위해 아직 Gmail이나 캐글 계정이 없다면 미리 생성해주세요!

- 4) Gmail ([가입 링크](#))
- 5) Kaggle : ([가입 링크](#))

## 03. 머신러닝이란?

### ▼ 6) 알고리즘이란?



#### 알고리즘

수학과 컴퓨터 과학, 언어학 또는 관련 분야에서 어떠한 문제를 해결하기 위해 정해진 일련의 절차나 방법을 공식화한 형태로 표현한 것, 계산을 실행하기 위한 단계적 절차  
- 위키피디아

어떤 문제를 풀기 위해 수학 공식을 만들었다고 보면 됩니다. 여기서 문제라는 건 다음과 같은 것이 될 수 있어요.

- A. 시험 전 날 커피를 몇 잔 마시면, 다음 날 시험에서 몇 점을 받을 수 있을까?
- B. 오늘 온도와 습도 데이터를 활용하여, 내일 미세먼지 농도를 예측할 수 있을까?
- C. 이 사진에 찍힌, 사람 수는 몇 명 일까?

일단 문제 A 부터 보죠!

▼ 문제 A. 시험 전 날 커피를 몇 잔 마시면, 다음 날 시험에서 몇 점을 받을 수 있을까?

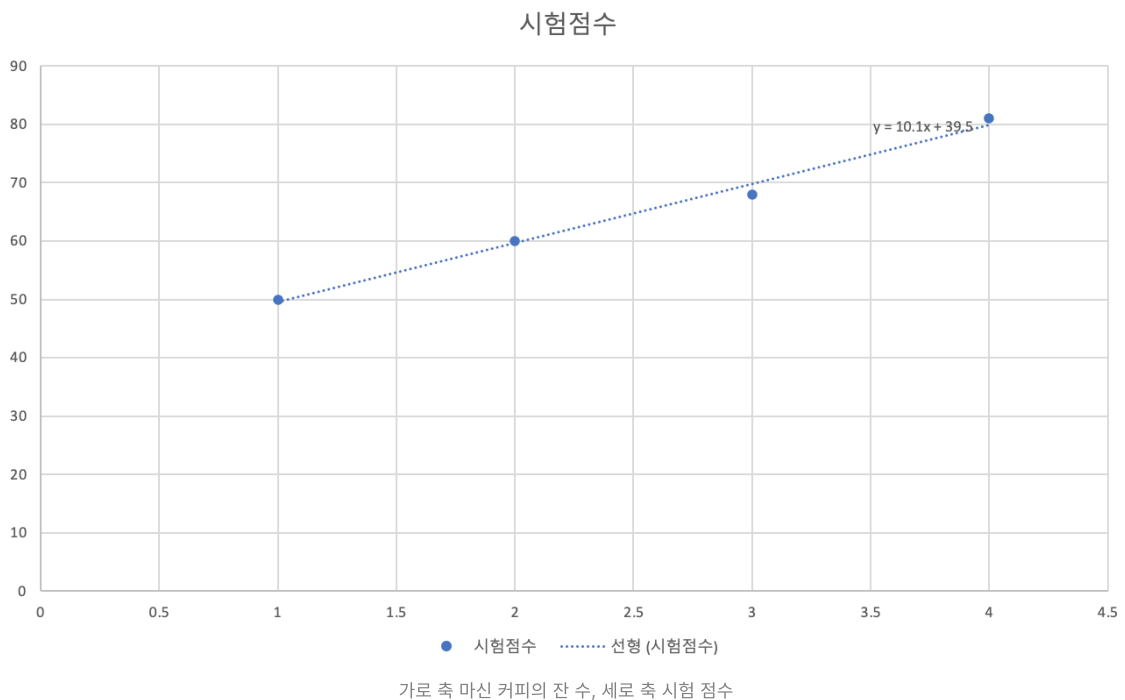
**?** 내일은 대망의 기말고사 시험 시작일입니다. 오늘 몇 잔의 커피를 마셔야 내일 시험 점수가 잘 나오는지 알고 싶습니다. 그래서 당신은 다음과 같은 실험을 했습니다.  
> 시험 전 날 마신 커피 잔 수와 그 다음 날 받은 시험 점수를 기록한다.

커피 잔 수에 따른 당신의 시험 점수

Aa 구분	# 커피 잔 수	# 시험 점수
제목 없음	1	50
제목 없음	2	60
제목 없음	3	68
제목 없음	4	81

만약 시험 전 날 5잔을 마시면 당신은 몇 점을 받을까요? 우리는 어렵지 않게 90점 정도를 받으리라는 것을 알 수 있습니다.

이것을 그래프로 표현하면 다음과 같아요.



수학자들은 이걸 공식화해서 "전날 마신 커피 잔 수에 따른 시험 점수"라는 알고리즘으로 만들고 싶었습니다. 그래서 다음과 같은 공식을 만들게 되죠.

$$\text{score} = 10 * \text{coffee} + 40$$

이 공식은 100% 정확하진 않았지만 추세를 예측하는데는 아주 큰 도움이 됐죠.

▼ 문제 B. 오늘 온도와 습도 데이터를 활용하여, 내일 미세먼지 농도를 예측할 수 있을까?

? 오늘 온도와 습도 데이터를 활용하여, 내일 미세먼지 농도를 예측할 수 있을까?

문제가 더 어려워졌습니다. 문제 A는 커피 잔 수만 고려하면 됐지만 이번에는 온도와 습도를 모두 고려해야 했어요. 수학자들은 머리가 아팠습니다.

이 문제를 어찌어찌 풀었다고 해도 더 큰 문제는 미세먼지 농도에 영향을 주는 요소는 온도와 습도 뿐만 아니라 바람의 방향, 요일, 계절, 시간 등에 따라 천차만별이었기 때문에 사람이 계산해서는 예측하기가 힘들었어요. 그래서 수학자들과 과학자들은 생각했습니다.

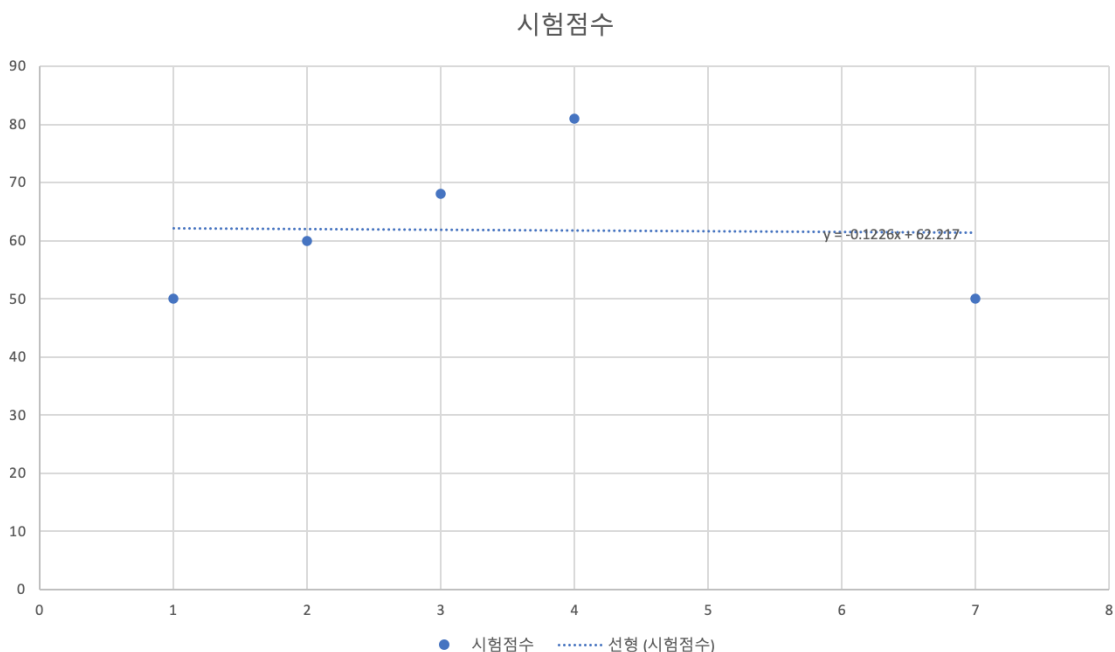
? 이런 문제를 컴퓨터가 풀게 할 수 있지 않을까?

이런 복잡한 문제를 손쉽게 풀어내기 위해 머신러닝이 생겨나게 되었습니다.

▼ 문제 A-1. 마신 커피 잔 수와 점수의 관계가 더 복잡하다면?

💡 더 복잡해진 문제는 딥러닝이 간단하게 해결할 수 있었습니다. 어디 한 번 살펴볼까요?

당신의 예측대로 시험 전 날 5잔의 커피를 마시면 90점을 맞을 수 있었습니다. 욕심이 난 당신은 커피를 7잔을 마셨습니다. 7잔을 마시면 확실하게 만점 100점을 받을 수 있을거라 생각했죠. 그러나 당신의 예상은 철저히 빗나갔습니다. 커피를 7잔이나 마신 당신은 잠을 한숨도 잘 수 없었고, 50점을 받고 말았습니다.



7잔 때의 데이터를 더하니 그래프도 이상해지고 수식도 이상해졌습니다. 수학자들은 고뇌하기 시작했죠. 직선 하나만 가지고는 풀 수 없는 문제가 있구나.

수학자와 데이터 과학자들은 협심하여 머신러닝을 연구하기 시작했어요. 1990년대부터 2000년대에 붐이 일었던 빅데이터 기술과 맞물려 머신러닝 연구의 추세는 딥러닝이 이끄는 방향으로 기울기 시작했습니다.



정확히 말하면 딥러닝은 머신러닝 방법 중 하나입니다. 머신러닝(기계학습)이라는 포괄적인 범위 안에 딥러닝(Deep learning) 깊은? 학습)이 포함되어 있죠.

연구 초반에는 MLP(Multi-Layer Perceptron)이라고 불렸으나 사람들의 유행을 타기 시작하면서 어감이 좋은 딥러닝으로 굳어지게 되었습니다.

그래서 머신러닝 중에서도 고차원의 비선형 문제를 잘 풀 수 있는 딥러닝 기술이 발전하기 시작했습니다. 딥러닝은 위에서 언급한 A, B, C 문제를 전부 잘 해결할 수 있게 되어 빠른 속도로 발전하였고 2000년대 초반에 이르러 전성기를 맞이하게 되었습니다.

요즘들어 우리가 말하는 인공지능과 머신러닝이라고 하는 것은, 여러 방법 중에서 월등하게 성능이 좋은 딥러닝을 말합니다. 최근 10년간 모든 학문 분야 중 가장 많은 연구가 이루어졌고 덕분에 파이썬이라는 언어가 덩달아 전성기를 맞게 되었죠.

우리는 이 딥러닝을 사용하기 위해 파이썬의 기초를 공부하고 딥러닝 기술 응용의 핵심적인 부분을 쉽고 빠르게 배워볼거예요!

#### ▼ 7) 머신러닝의 회귀와 분류



머신러닝에서 문제를 풀 때, 해답을 내는 방법을 크게 **회귀** 또는 **분류**로 나눌 수 있습니다. 각각이 무엇을 의미하는지 배워봅시다!

##### ▼ 회귀 (Regression)

아래와 같은 문제를 푼다고 가정해봅시다.



사람의 얼굴 사진을 보고 몇 살인지 예측하는 문제

모든 문제를 풀기 위해서는 먼저 입력값(Input)과 출력값(Output)을 정의해야 합니다. 이 문제에서 입력값은 [얼굴 사진]이 되고 출력값은 [예측한 나이]가 됩니다.

이 때 출력값인 나이를 소수점(float)로 표현하여 아래와 같이 표현하도록 할 수 있습니다.

##### 사진으로 나이 예측 (회귀)

Aa 사람	@ 얼굴 사진	≡ 나이
A		24.0
B		21.0
C		82.0
D		45.0

나이의 값은 연속적이죠. 예를들어 1살, 2살, 3살, ..., 15살, 16살, ..., 89살, 90살... 이렇게 계속해서 나열할 수 있습니다. 이런 식으로 출력값이 연속적인 소수점으로 예측하게 하도록 푸는 방법을 회귀라고 합니다.

##### ▼ 분류 (Classification)

분류 문제는 좀 더 쉬워요. 아래의 문제를 예를 들어보죠!



대학교 시험 전 날 공부한 시간을 가지고 해당 과목의 이수 여부(Pass or fail)를 예측하는 문제

이 문제에서 입력값은 [공부한 시간] 그리고 출력값은 [이수 여부]가 됩니다. 우리는 이수 여부를 0, 1 이라는 이진 클래스(Binary class)로 나눌 수 있습니다. 0이면 미이수(Fail), 1이면 이수(Pass) 이런식으로요. 이런 경우를 이진 분류(Binary classification)이라고 부릅니다.

##### 성적 - 이진 분류

Aa 이수 여부	# 클래스
이수 (Pass)	0

Aa 이수 여부	# 클래스
미이수 (Fail)	1

아래 문제는 어떨까요?

**?** 대학교 시험 전 날 공부한 시간을 가지고 해당 과목의 성적(A, B, C, D, F)을 예측하는 문제

예상하셨겠지만 아래와 같이 클래스를 5개의 클래스로 나누고 이 방법을 **다중 분류**(Multi-class classification, Multi-label classification)라고 부릅니다.





#### 성적 - 다중 분류

Aa 성적	# 클래스
A	0
B	1
C	2
D	3
E	4

#### ▼ 회귀와 분류 둘 다 가능한 문제

몇몇 문제는 회귀/분류 두 가지 방법으로 접근하여 해결할 수 있어요. 예를 들어 얼굴 사진을 보고 나이를 예측하는 문제를 다시 떠올려보죠. 아래와 같이 회귀 방법으로 문제를 풀 수도 있지만,

#### 사진으로 나이 예측 (회귀)

Aa 사람	🖼️ 얼굴 사진	≡ 나이
A		24.0
B		21.0
C		82.0
D		45.0





만약 나이를 범위로 쪼개어 생각하면 어떨까요?

#### 나이를 클래스로!

Aa 나이 범위	# 클래스
0 - 9세	0
10 - 19세	1
20 - 29세	2
...	
90 - 99세	9
100 - 109세	10

우리는 나이 범위에 따른 클래스로 나누어서 생각할 수 있습니다. 이렇게 되면 다중 분류 문제로 바꾸어서 풀 수 있게 되죠! 다중 분류 문제로 얼굴 사진을 보고 나이를 예측하는 문제를 풀어보면 아래와 같이 될 겁니다.

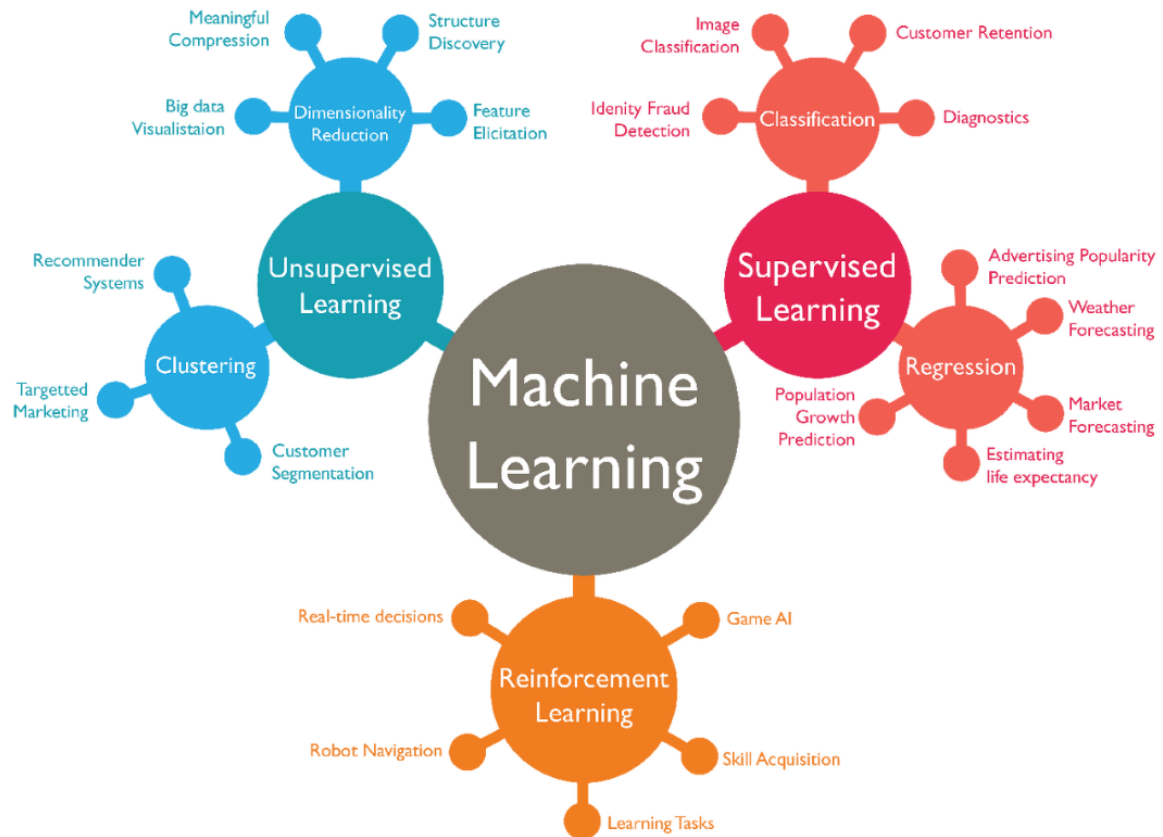
#### 사진으로 나이 예측 (분류)

Aa 사람	🖼️ 얼굴 사진	≡ 나이	# 클래스
A		24.0	2
B		21.0	2
C		82.0	8
D		45.0	4

#### ▼ 8) 지도 학습/비지도 학습/강화 학습



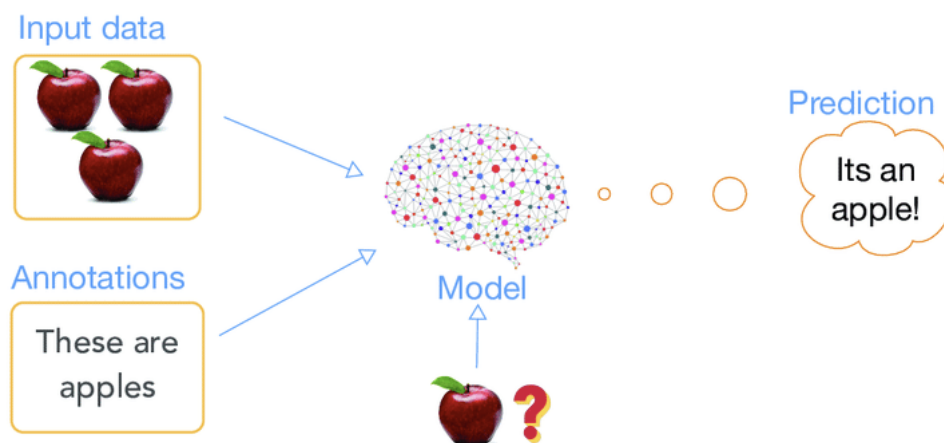
머신러닝은 크게 3가지로 분류해요. 바로 지도/비지도/강화 학습이죠. 각각의 개념과 차이점에 대해서 알아보까요?



출처: <https://towardsdatascience.com/the-future-with-reinforcement-learning-877a17187d54>

▼ 지도 학습(Supervised learning): 정답을 알려주면서 학습시키는 방법

## supervised learning

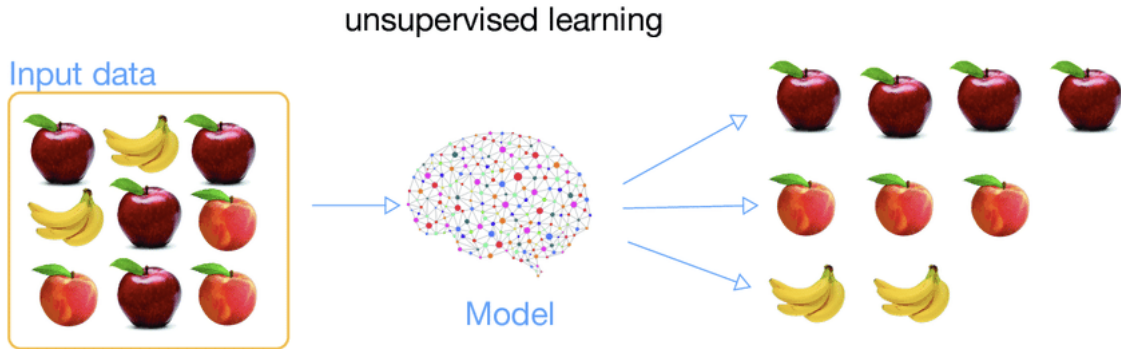


출처: [https://www.researchgate.net/figure/Supervised-learning-and-unsupervised-learning-Supervised-learning-uses-annotation\\_fig1\\_329533120](https://www.researchgate.net/figure/Supervised-learning-and-unsupervised-learning-Supervised-learning-uses-annotation_fig1_329533120)

우리가 위에서 배웠던 회귀와 분류 문제가 대표적인 지도 학습에 속합니다. 지도 학습은 기계에게 입력값과 출력값을 전부 보여주면서 학습시키죠. 우리는 이미 정답을 알고 있기 때문에 정답을 잘 맞추었는지 아닌지를 쉽게 파악할 수 있습니다. 대신 정답(출력값)이 없으면 이 방법으로 학습시킬 수 없습니다.

여러분이 회사에서 머신러닝 엔지니어로 근무한다면 회사에서 필요로 하는 문제를 풀기 위해 많은 데이터를 필요로 할 겁니다. 근데 대부분 회사에는 데이터가 없는 경우가 많죠. 혹은 입력값에 해당하는 데이터는 있어도 출력값(정답)에 해당하는 데이터가 없는 경우가 비일비재합니다. 따라서 입력값에 정답을 하나씩 입력해주는 작업을 하게 되는 경우가 있는데 그 과정을 **노가다 라벨링 (Labeling, 레이블링)** 또는 **어노테이션(Annotation)**이라고 합니다.

▼ 비지도 학습 (Unsupervised learning): 정답을 알려주지 않고 군집화(Clustering)하는 방법



출처: [https://www.researchgate.net/figure/Supervised-learning-and-unsupervised-learning-Supervised-learning-uses-annotation\\_fig1\\_329533120](https://www.researchgate.net/figure/Supervised-learning-and-unsupervised-learning-Supervised-learning-uses-annotation_fig1_329533120)

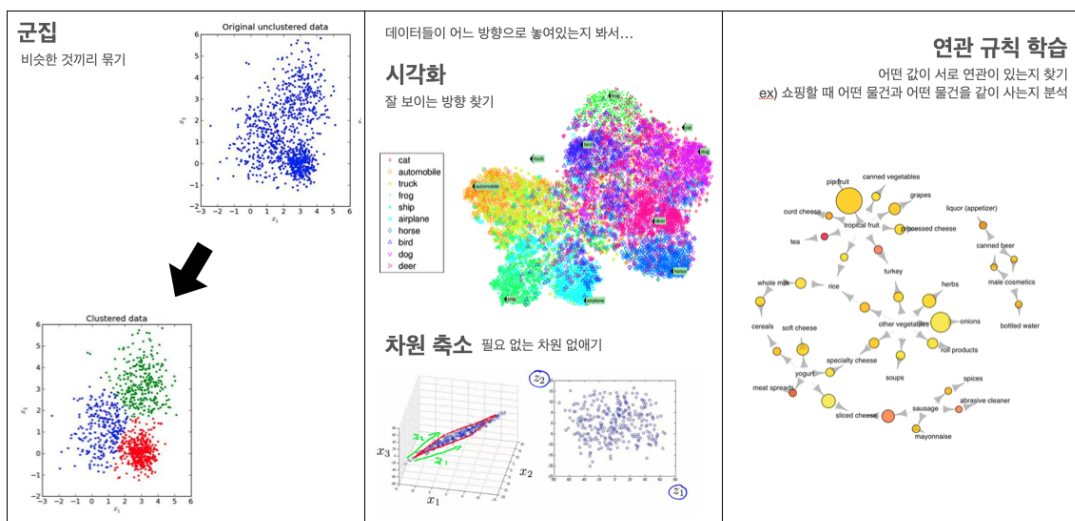
비지도 학습은 그룹핑 알고리즘(Grouping algorithm)의 성격을 띄고 있습니다. 예를 들어 음악을 분석하여 장르를 구분하는 문제를 푼다고 가정해봅시다.

? 음원 파일을 분석하여 장르를 팝, 락, 클래식, 댄스로 나누는 문제

우리가 가지고 있는 데이터에 입력값(음원파일)과 출력값(장르) 둘 다 존재한다면 우리는 지도 학습으로 이 문제를 풀 수 있지만 출력값에 해당하는 장르 데이터가 없을 때 비지도 학습 방법을 사용합니다. 비지도 학습 방법은 라벨(Label 또는 Class)이 없는 데이터를 가지고 문제를 풀어야 할 때 큰 힘을 발휘하죠! 음악 장르를 구분하는 문제를 비지도 학습을 사용해서 풀라고 시키면 아래와 같은 뉘앙스가 됩니다.

👉 우리에게 수 백만개의 음원 파일이 있는데, 각 음원 파일에 대한 장르 데이터는 없어. 그러니까 기계에게 음원 파일을 들려주고 알아서 비슷한 것끼리 분류하게 해보자!

▼ 비지도 학습의 종류 (참고)

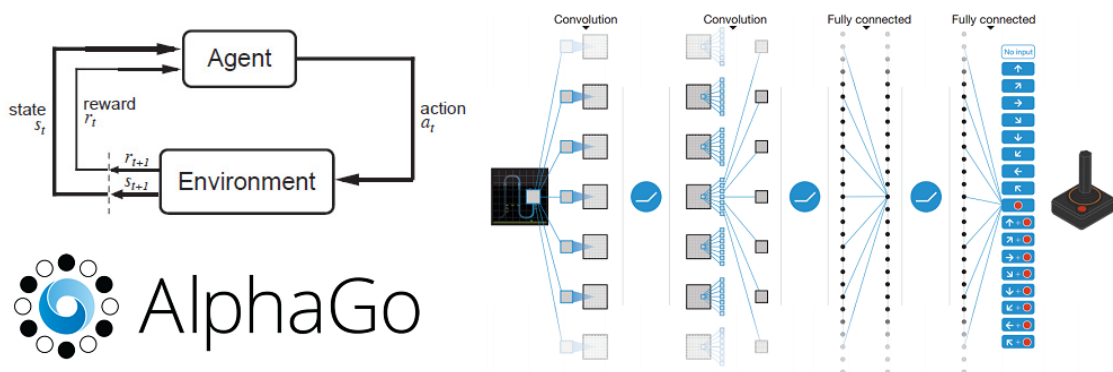


- 군집 (Clustering)
  - K-평균 (K-Means)



- 계층 군집 분석(HCA, Hierarchical Cluster Analysis)
- 기댓값 최대화 (Expectation Maximization)
- 시각화(Visualization)와 차원 축소(Dimensionality Reduction)
  - 주성분 분석(PCA, Principal Component Analysis)
  - 커널 PCA(Kernel PCA)
  - 지역적 선형 임베딩(LLE, Locally-Linear Embedding)
  - t-SNE(t-distributed Stochastic Neighbor Embedding)
- 연관 규칙 학습(Association Rule Learning)
  - 어프라이어리(Apriori)
  - 이클렛(Eclat)

▼ 강화 학습(Reinforcement learning): 주어진 데이터없이 실행과 오류를 반복하면서 학습하는 방법 (알파고를 탄생시킨 머신러닝 방법!!)



출처: <https://adeshpande3.github.io/Deep-Learning-Research-Review-Week-2-Reinforcement-Learning>.

행동 심리학에서 나온 이론으로 분류할 수 있는 데이터가 존재하지 않거나, 데이터가 있어도 정답이 따로 정해져 있지 않고, 자신이 한 행동에 대해 보상(Reward)를 받으며 학습하는 것을 말합니다.

- 강화학습의 개념
  - 에이전트(Agent)
  - 환경(Environment)
  - 상태(State)
  - 행동(Action)
  - 보상(Reward)

게임을 예로 들면 게임의 규칙을 따로 입력하지 않고 자신(Agent)이 게임 환경(Environment)에서 현재 상태(State)에서 높은 점수(Reward)를 얻는 방법을 찾아가며 행동(Action)하는 학습 방법으로 특정 학습 횟수를 초과하면 높은 점수(Reward)를 획득할 수 있는 전략이 형성되게 됩니다. 단, 행동(Action)을 위한 행동 목록(방향키, 버튼)들은 사전에 정의가 되어야 합니다.

[DEVSISTERS 머신러닝 팀] 딥러닝과 강화 학습으로 학습한 쿠키런 AI

만약 이것을 지도 학습(Supervised Learning)의 분류(Classification)를 통해 학습을 한다고 가정하면 모든 상황에 대해 어떠한 행동을 해야 하는지 모든 상황을 예측하고 답을 설정해야 하기 때문에 엄청난 데이터가 필요합니다. 예를 들어 바둑을 학습한다고 했을 때, 지도 학습(Supervised Learning)의 분류(Classification)를 이용해 학습하는 경우 아래와 같은 개수의 데이터가 필요해지게 됩니다.



바둑의 경우의 수

208168199381979984699478633344862770286522453884530548425639456820927419612738015:

강화 학습(Reinforcement learning)은 이전부터 존재했던 학습법이지만 이전에 알고리즘은 실생활에 적용할 수 있을 만큼 좋은 결과를 내지 못했습니다. 하지만 딥러닝의 등장 이후 강화 학습에 신경망을 적용하면서부터 바둑이나 자율주행차와 같은 복잡

한 문제에 적용할 수 있게 되었습니다.

## 04. 선형 회귀 (Linear Regression)

### ▼ 9) 선형 회귀와 가설, 손실 함수 Hypothesis & Cost function (Loss function)

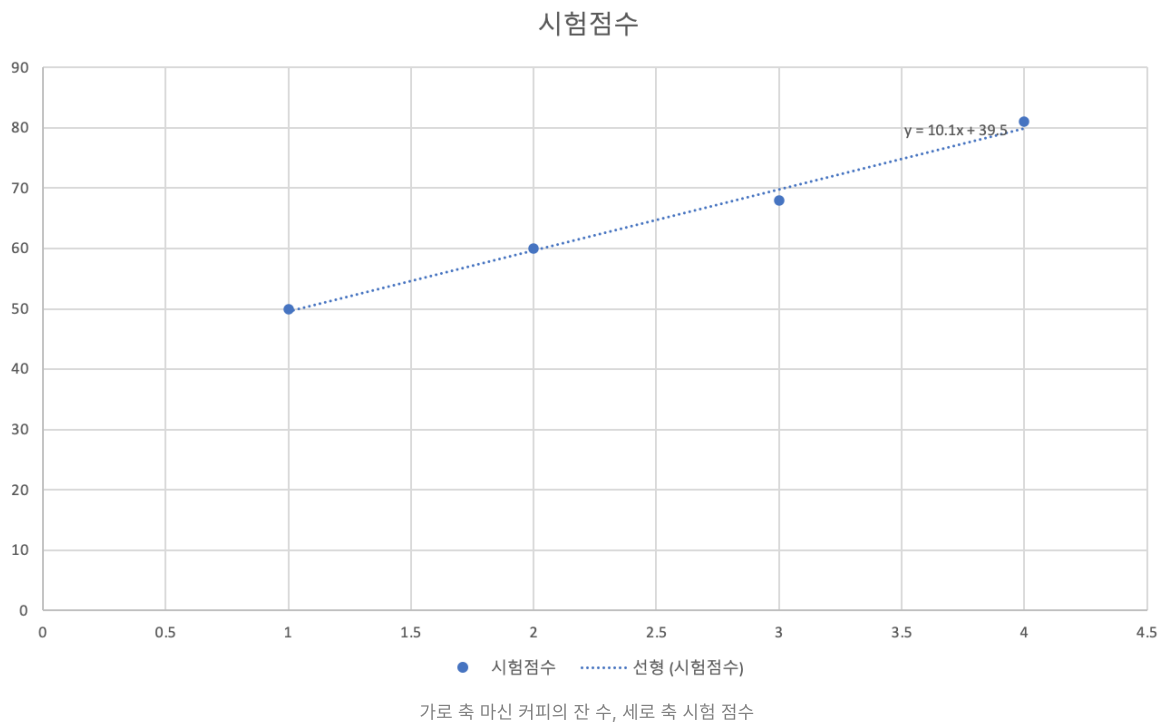
아래와 같은 문제를 가정해봅시다!

? 시험 전 날 마신 커피 잔 수에 따라 시험 점수를 예측할 수 있을까?

커피 잔 수에 따른 당신의 시험 점수

Aa 구분	# 커피 잔 수	# 시험 점수
제목 없음	1	50
제목 없음	2	60
제목 없음	3	68
제목 없음	4	81

우리는 위의 표와 같은 데이터가 있고 그것을 아래와 같이 그래프로 표시할 수 있습니다.



우리는 이 그래프를 보고 가설을 세울 수 있는데요. 임의의 직선 1개로 이 그래프를 비슷하게 표현할 수 있다고 가설을 세울 수 있습니다. 이 선형 모델은 수식으로 아래와 같이 표현할 수 있어요. (직선 = 1차 함수)

$$H(x) = Wx + b$$

우리는 정확한 시험 점수를 예측하기 위해 우리가 만든 임의의 직선(가설)과 점(정답)의 거리가 가까워지도록 해야합니다. (=mean squared error)

$$Cost = \frac{1}{N} \sum_{i=1}^N (H(x_i) - y_i)^2$$

여기서  $H(x)$ 는 우리가 가정한 직선이고  $y$ 는 정답 포인트라고 했을 때  $H(x)$ 와  $y$ 의 거리(또는 차의 절대값)가 최소가 되어야 이 모델이 잘 학습되었다고 말할 수 있을겁니다.

여기서 우리가 임의로 만든 직선  $H(x)$ 를 가설(Hypothesis)이라고 하고 Cost를 손실 함수(Cost or Loss function)라고 합니다.



### 머신러닝 더 알아보기!

수학은 여기서 끝! 가장 기본적인 1차 함수로 머신러닝을 설명했습니다. 실무에서 사용하는 머신러닝 모델은 1차 함수보다 더 높은 고차원 함수이지만 원리는 똑같습니다.

우리는 데이터를 보고 "어떤 함수에 비슷한 모양일 것이다"라고 가설을 세우고 그에 맞는 손실 함수를 정의합니다. 우리가 하는 일은 여기서 끝나고 기계는 우리가 정의한 손실 함수를 보고 우리의 가설에 맞출 수 있도록 열심히 계산하는 일을 합니다. 그래서 기계학습(머신러닝)이라는 이름이 붙게 된 것이지요.

#### ▼ 10) 다중 선형 회귀(Multi-variable linear regression)

선형 회귀와 똑같지만 입력 변수가 여러개라고 생각하면 쉬워요. 예를 들면 아래 표와 같죠.

커피 잔 수와 게임 플레이 시간 따른 당신의 시험 점수

Aa 구분	# 커피 잔 수	# 게임 플레이 시간	# 시험 점수
제목 없음	1	3	45
제목 없음	2	4	52
제목 없음	3	5	33
제목 없음	4	2	81
제목 없음	5	1	95

위에서는 마신 커피 잔 수만 입력값으로 들어갔지만 만약 입력값이 2개 이상이 되는 문제를 선형 회귀로 풀고 싶을 때 다중 선형 회귀를 사용합니다.

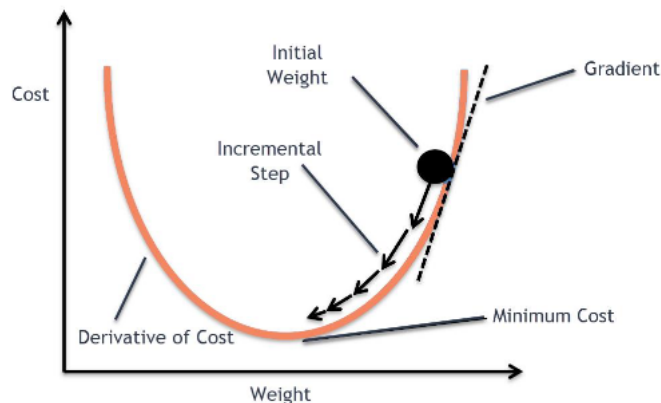
$$\text{가설 } H(x_1, x_2, \dots, x_n) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

$$\text{손실 함수 } Cost = \frac{1}{N} \sum (H(x_1, x_2, x_3, \dots, x_n) - y)^2$$

## 05. 경사 하강법 (Gradient descent method)

#### ▼ 11) 경사 하강법이란?

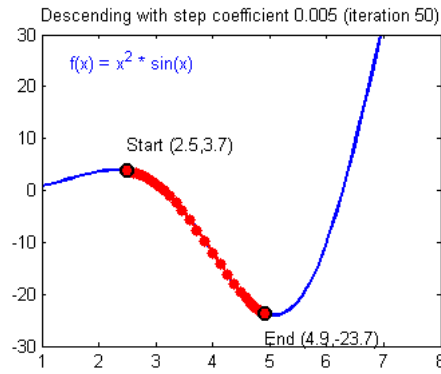
우리는 위에서 정의한 손실 함수를 대충 상상해 볼겁니다. 손실 함수가 아래와 같은 모양을 가지고 있다고 가정해보죠.



출처: <https://towardsdatascience.com/using-machine-learning-to-predict-fitbit-sleep-scores-496a7d9ec48>

우리의 목표는 손실 함수를 최소화(Optimize)하는 것입니다. 손실 함수를 최소화하는 방법은 이 그래프를 따라 점점 아래로 내려가야 하죠.

컴퓨터는 사람처럼 수식을 풀 수 없기 때문에 경사 하강법이라는 방법을 써서 점진적으로 문제를 풀어갑니다. 처음에 랜덤으로 한 점으로부터 시작합니다. 좌우로 조금씩 그리고 한번씩 움직이면서 이전 값보다 작아지는지를 관찰합니다. 한칸씩 전진하는 단위를 Learning rate라고 부르죠. 그리고 그래프의 최소점에 도달하게 되면 학습을 종료하면 되겠죠.

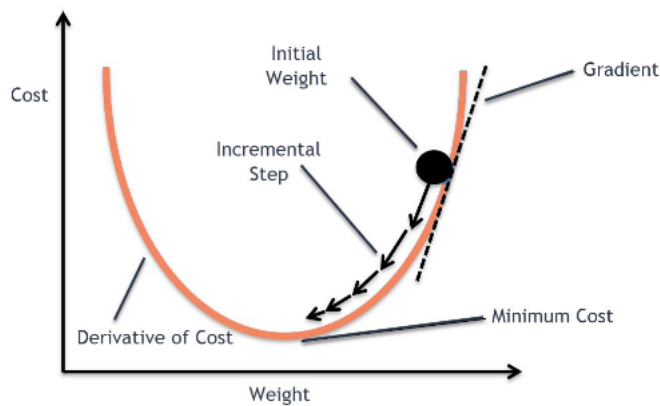


출처: <https://medium.com/hackernoon/life-is-gradient-descent-880c60ac1be8>

## ▼ 12) Learning rate



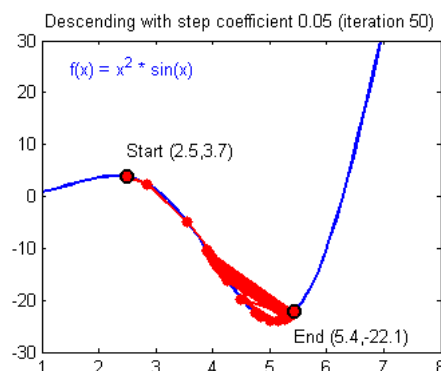
우리가 만든 머신러닝 모델이 학습을 잘하기 위해서는 적당한 Learning rate를 찾는 노가다가 필수적입니다!



출처: <https://towardsdatascience.com/using-machine-learning-to-predict-fitbit-sleep-scores-496a7d9ec48>

위의 그래프에서 만약 Learning rate가 작다면 어떻게 될까요? 초기 위치로부터 최소점을 찾는 데까지 많은 시간이 걸릴 것입니다. 그리고 이 것은 학습하는데, 즉 최소값에 수렴하기까지 많은 시간이 걸린다는 것을 뜻합니다.

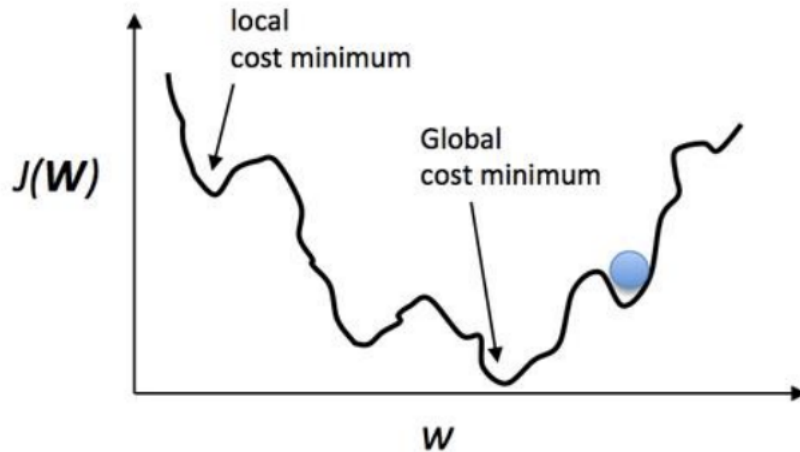
반대로 만약 Learning rate가 지나치게 크다면요? 우리가 찾으려는 최소값을 지나치고 검은 점은 계속 진동하다가 최악의 경우에는 발산하게 될 수도 있습니다. 이런 상황을 Overshooting이라고 부릅니다.



출처: <https://medium.com/hackernoon/life-is-gradient-descent-880c60ac1be8>

## ▼ 13) 실제로 손실 함수를 그릴 수 있을까?

간단한 선형 회귀 문제의 경우는 그래프를 그릴 수는 있지만 복잡한 가설을 세울 경우에는 사람이 그릴 수도 없고 상상할 수 없는 형태가 됩니다. 예를 들어 간단하게 아래와 같은 그래프를 상상할 수 있겠군요.



출처: <https://regenerativetoday.com/logistic-regression-with-python-and-scikit-learn/>

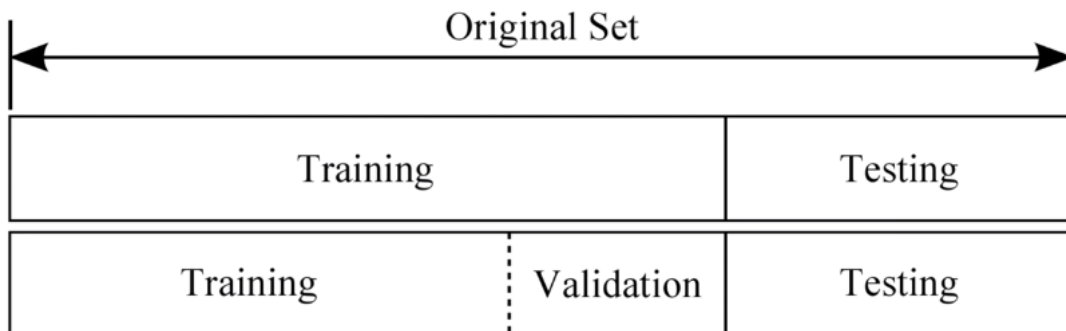
우리의 목표는 이 손실 함수의 최소점인 Global cost minimum을 찾는 것입니다. 그런데 우리가 한 칸씩 움직이는 스텝(Learning rate)을 잘못 설정할 경우 Local cost minimum에 빠질 가능성이 높습니다. Cost가 높다는 얘기는 우리가 만든 모델의 정확도가 낮다는 말과 같지요. 따라서 우리는 최대한 Global minimum을 찾기 위해 좋은 가설과 좋은 손실 함수를 만들어서 기계가 잘 학습할 수 있도록 만들어야하고 그것이 바로 머신러닝 엔지니어의 핵심 역할입니다!

## 06. 데이터셋 분할

### ▼ 14) 학습/검증/테스트 데이터



각각의 데이터가 어떤 용도로 사용되는지 알아볼까요?



출처: <https://3months.tistory.com/118>

#### 1. **Training set** (학습 데이터셋, 트레이닝셋) = 교과서

머신러닝 모델을 학습시키는 용도로 사용합니다. 전체 데이터셋의 약 80% 정도를 차지합니다.

#### 2. **Validation set** (검증 데이터셋, 밸리데이션셋) = 모의고사

머신러닝 모델의 성능을 검증하고 튜닝하는 지표의 용도로 사용합니다. 이 데이터는 정답 라벨이 있고, 학습 단계에서 사용하기는 하지만, 모델에게 데이터를 직접 보여주지는 않으므로 모델의 성능에 영향을 미치지 않습니다.

손실 함수, Optimizer 등을 바꾸면서 모델을 검증하는 용도로 사용합니다. (실습에서 자세히 다룰게요!)

전체 데이터셋의 약 20% 정도를 차지합니다.

#### 3. **Test set** (평가 데이터셋, 테스트셋) = 수능

정답 라벨이 없는 실제 환경에서의 평가 데이터셋입니다. 검증 데이터셋으로 평가된 모델이 아무리 정확도가 높더라도 사용자가 사용하는 제품에서 제대로 동작하지 않는다면 소용이 없겠죠?

## 07. 실습 환경 소개 (Colab)

### ▼ 15) 실습 환경 소개 (Colab)



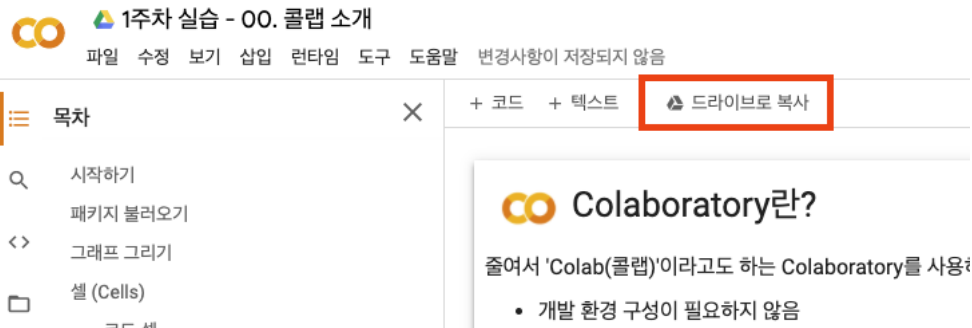
#### Colaboratory이란?

구글에서 만든 개발 환경으로, 구글 드라이브에 파일을 올려놓고 웹 상에서 직접 코드를 돌릴 수 있어요! 우선 직접 코드를 실행해보면서 친해져볼까요? 😊

### ▼ [코드스니펫] - colab 실습 링크

<https://colab.research.google.com/drive/1qfFR5qiAjaqpDuERWeNd1KJWpo0vY7ej?usp=sharing>

[드라이브로 복사] 버튼을 눌러 여러분의 구글드라이브에 Colab 노트북을 저장합니다. 그리고 사본으로 만든 Colab 노트북에서 따라하세요!



- Colab은 Jupyter Notebook을 기반으로 하고 있어, 각 '메모장'은 셀로 이루어져 있고, 이 셀 단위로 코드를 실행하고 그 결과를 출력해볼 수 있습니다.
- 데이터분석/머신러닝에 자주 쓰이는 다양한 패키지들이 기본적으로 설치되어 있고, 필요에 따라 추가적으로 더 설치해줄 수도 있습니다.

## 08. 간단한 선형회귀 실습

### ▼ 16) 선형회귀 실습

- 우선 TensorFlow를 이용하여 직접 선형회귀 코드를 실행해봅시다!

### ▼ [코드스니펫] - 선형회귀 실습

<https://colab.research.google.com/drive/1FksWe-MrejLcc4e4psqBAK9io8m3UYKo?usp=sharing>

1. 텐서플로를 임포트하고 데이터와 변수들을 설정해줍니다.

```
import tensorflow as tf

tf.compat.v1.disable_eager_execution()

x_data = [[1, 1], [2, 2], [3, 3]]
y_data = [[10], [20], [30]]

X = tf.compat.v1.placeholder(tf.float32, shape=[None, 2])
Y = tf.compat.v1.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random.normal(shape=(2, 1)), name='W')
b = tf.Variable(tf.random.normal(shape=(1,)), name='b')
```

2. 가설과 비용함수, optimizer를 정의합니다.

```
hypothesis = tf.matmul(X, W) + b
cost = tf.reduce_mean(tf.square(hypothesis - Y))
optimizer = tf.compat.v1.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)
```

3. 매 스텝 별로 결과를 출력하며 비용함수가 줄어드는 것을 확인합니다.

```
with tf.compat.v1.Session() as sess:
    sess.run(tf.compat.v1.global_variables_initializer())

    for step in range(50):
        c, W_, b_ = sess.run([cost, W, b, optimizer], feed_dict={X: x_data, Y: y_data})
        print('Step: %2d\t loss: %.2f\t' % (step, c))

    print(sess.run(hypothesis, feed_dict={X: [[4, 4]]}))
```

• 실제로는 이렇게 Keras를 이용해 쉽게 선형회귀를 실행해볼 수 있습니다!

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam, SGD

x_data = np.array([[1], [2], [3]])
y_data = np.array([[10], [20], [30]])

model = Sequential([
    Dense(1)
])

model.compile(loss='mean_squared_error', optimizer=SGD(lr=0.1))

model.fit(x_data, y_data, epochs=100) # epochs 복수형으로 쓰기!
```

```
y_pred = model.predict([[4]])
print(y_pred)
```

## 09. 캐글 선형회귀 실습

### ▼ 17) Kaggle 데이터 선형회귀 실습



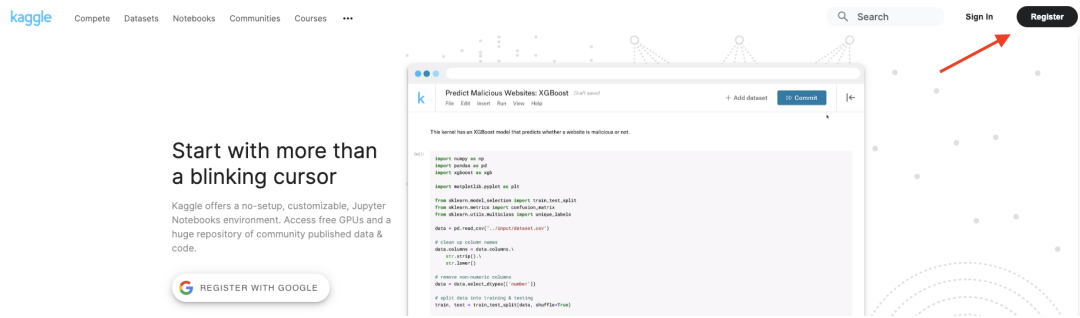
Kaggle은 데이터 사이언티스트를 위한 커뮤니티입니다! 다양한 데이터셋이 공개되어 있어 직접 분석해보고 결과를 공유하고 서로 비교해볼 수도 있고, 기업 및 단체에서 올린 문제를 풀어 상품을 받을 수도 있습니다. 우리는 Kaggle에 공개되어 있는 데이터셋을 가져와 실습을 해보겠습니다.

### ▼ [코드스니펫] - Kaggle 데이터로 실습하기

<https://colab.research.google.com/drive/1Jqn60TFKZ46STP6c5ZlQqQFBDLJyfa1K?usp=sharing>

▼ 우선 콜랩에서 캐글 데이터셋을 직접 다운로드 받는 법을 배워보겠습니다.

#### 1. 캐글 회원가입하기



2. 로그인하고 내 프로필 클릭 > Account 탭 클릭하기
3. API - Create New API Token 클릭해서 kaggle.json 파일 다운로드 받기
4. 브라우저에서 파일을 열어 username과 key 복사하기
5. 환경변수 지정하기

```
import os
os.environ['KAGGLE_USERNAME'] = '[내_캐글_username]' # username
os.environ['KAGGLE_KEY'] = '[내_캐글_key]' # key
```

6. 원하는 데이터셋의 API를 복사해 와 실행하기

```
!kaggle datasets download -d ashydv/advertising-dataset
```

7. 데이터셋 압축 풀어주기

```
!unzip /content/advertising-dataset.zip
```

▼ 데이터를 받아왔으니, 이제 본격적으로 분석해보아야겠죠!

1. 필요한 라이브러리들 임포트하기

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam, SGD
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

2. 데이터셋 불러와서 형태 확인하기

```
df = pd.read_csv('advertising.csv')
df.head(5)
```

```
print(df.shape)
```

3. 데이터셋 살짝 살펴보기

```
sns.pairplot(df, x_vars=['TV', 'Newspaper', 'Radio'], y_vars=['Sales'], height=4)
```

4. 데이터셋 가공하기

```
x_data = np.array(df[['TV']], dtype=np.float32)
y_data = np.array(df['Sales'], dtype=np.float32)
```



```
print(x_data.shape)
print(y_data.shape)
```

```
x_data = x_data.reshape((-1, 1))
y_data = y_data.reshape((-1, 1))

print(x_data.shape)
print(y_data.shape)
```

## 5. 데이터셋을 학습 데이터와 검증 데이터로 분할하기

```
x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.2, random_state=2021)

print(x_train.shape, x_val.shape)
print(y_train.shape, y_val.shape)
```

## 6. 학습시키기

```
model = Sequential([
    Dense(1)
])

model.compile(loss='mean_squared_error', optimizer=Adam(lr=0.1))

model.fit(
    x_train,
    y_train,
    validation_data=(x_val, y_val), # 검증 데이터를 넣어주면 한 epoch이 끝날때마다 자동으로 검증
    epochs=100 # epochs 복수형으로 쓰기!
)
```

## 7. 검증 데이터로 예측하기

```
y_pred = model.predict(x_val)

plt.scatter(x_val, y_val)
plt.scatter(x_val, y_pred, color='r')
plt.show()
```

## 8. 여러 X값을 이용하여 매출 예측하기

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam, SGD
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

df = pd.read_csv('advertising.csv')

x_data = np.array(df[['TV', 'Newspaper', 'Radio']], dtype=np.float32)
y_data = np.array(df['Sales'], dtype=np.float32)

x_data = x_data.reshape((-1, 3))
y_data = y_data.reshape((-1, 1))

print(x_data.shape)
print(y_data.shape)

x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.2, random_state=2021)

print(x_train.shape, x_val.shape)
print(y_train.shape, y_val.shape)

model = Sequential([
    Dense(1)
])


model.compile(loss='mean_squared_error', optimizer=Adam(lr=0.1))

model.fit(
```

```
x_train,
y_train,
validation_data=(x_val, y_val), # 검증 데이터를 넣어주면 한 epoch이 끝날때마다 자동으로 검증
epochs=100 # epochs 복수형으로 쓰기!
)
```

## 10. 1주차 끝 & 숙제 설명

▼ 혼자서 Linear regression 구현하기

 숙제로는 스스로 선형회귀를 구현해봅시다 😊  
연차로부터 연봉을 예측해보아요~

▼ [코드스니펫] - 숙제 데이터셋 다운로드 받기

```
!kaggle datasets download -d rsadiq/salary
!unzip salary.zip
```

- 연차-연봉 데이터셋 살펴보기 <https://www.kaggle.com/rsadiq/salary>
- Learning rate(lr)를 바꾸면서 실험하기
- Optimizer를 바꾸면서 실험하기
- 손실함수(loss)를 mean\_absolute\_error 로 바꿔서 실험하기

## 11. 1주차 숙제 답안 코드

▼ [코드스니펫] - 1주차 숙제 답안 코드

전체코드

▼ 코드

```
import os
os.environ['KAGGLE_USERNAME'] = 'username' # username
os.environ['KAGGLE_KEY'] = 'key' # key
```

```
!kaggle datasets download -d rsadiq/salary
```

```
!unzip salary.zip
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam, SGD
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

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

```
x_data = np.array(df['YearsExperience'], dtype=np.float32)
y_data = np.array(df['Salary'], dtype=np.float32)

x_data = x_data.reshape((-1, 1))
y_data = y_data.reshape((-1, 1))

print(x_data.shape)
```

```

print(y_data.shape)

x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.2, random_state=2021)

print(x_train.shape, x_val.shape)
print(y_train.shape, y_val.shape)

model = Sequential([
    Dense(1)
])

model.compile(loss='mean_squared_error', optimizer=SGD(lr=0.01))

model.fit(
    x_train,
    y_train,
    validation_data=(x_val, y_val), # 검증 데이터를 넣어주면 한 epoch이 끝날때마다 자동으로 검증
    epochs=100 # epochs 복수형으로 쓰기!
)

```

```

y_pred = model.predict(x_val)

plt.scatter(x_val, y_val)
plt.scatter(x_val, y_pred, color='r')
plt.show()

```

(<https://colab.research.google.com/drive/1R7LnFiRL0O8KvuQkCJCdbJ4Y9GXqjWhJ?usp=sharing#scrollTo=XtBmYcOHAVgQ>)

Copyright © TeamSparta All rights reserved.