



인공지능 vs 머신러닝

- 인공지능 연구의 흐름
- AI & 머신러닝 체험
- 머신러닝 종류
- 첫번째 알고리즘 k-NN

인공지능 연구의 주요 방향

➤ 기호주의 (Symbolism)

- AI를 인간의 논리적 사고와 비슷한 방식으로 구축하려는 접근 방식.
- 초기 AI 연구의 중심 방향이 되었음
- **명시적인 규칙과 논리 이용**하는 문제 해결에 강점, 유연한 문제 해결에는 한계 보임-모라벡의 역설
예) 전문가 시스템
- 대표학자: 존 매카시, 마빈 민스키, 앨런 뉴웰, 허버트 사이먼, 조지프 와이젠바움, 에드워드 페이젠바움, 존 맥더모트

➤ 연결주의 (Connectionism)

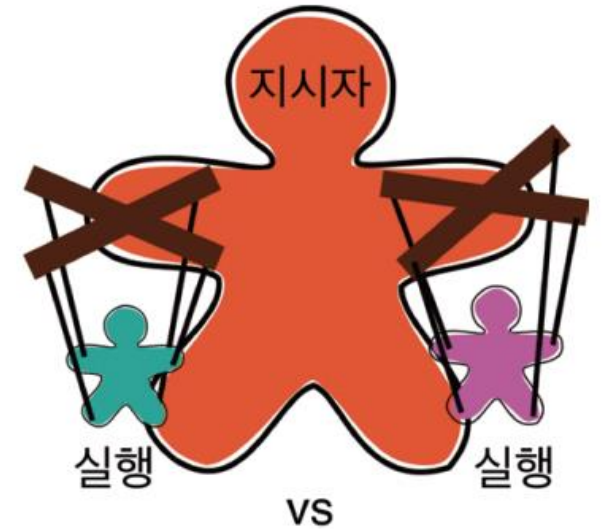
- 뇌의 뉴런과 유사한 구조를 모방한 인공 신경망을 통해 학습과 문제 해결을 시도하는 접근 방식
- 1960년대 퍼셉트론 연구에서 시작되어 1980년대 역전파 알고리즘의 도입으로 크게 발전
- 2000년대 중반 이후, 딥러닝 기술의 발전으로 다시 주목받게 되었고, 이미지 인식, 자연어 처리 등 여러 분야에서 뛰어난 성과를 거둠. 예) 알파고
- 대표학자: 프랭크 로젠블랫, 마빈 민스키, 시모어 페퍼트, 데이비드 럼멜하트, 제프리 힌튼, 알렉스 크리제브스키, 데미스 허사비스, 존 점퍼

인공지능 연구의 주요 방향

시기	기호주의 (Symbolism)	연결주의 (Connectionism)
1950년	- 앨런 튜링, "계산 기계와 지능" 논문 발표, 튜링 테스트 제안(이미테이션게임)	
1950년대	- 1956: 다트머스 회의에서 '인공지능' 용어 처음 사용.	
	- 기호주의 AI 출발	
1960~ 1970년대	- 논리와 추론을 통한 문제 해결 (제어 이론, 자동 정리 증명)	- 1960s: 퍼셉트론 연구 (로젠블랫)
	- 1965: ELIZA (자연어 처리 초기 시스템)	- 1969: 마빈 민스키와 시모어 페퍼트, 퍼셉트론 한계 지적
	- 1970s: 전문가 시스템 개발 (MYCIN 등)	
1980년대	- 1980s: 전문가 시스템 상업적 응용 시작 (XCON 등)	- 1980s: 역전파 알고리즘 제안 (럼멜하트, 힌튼), 신경망 연구 부활
1990년대	- 기계 학습 알고리즘 발전 (SVM, 결정 트리 등)	- 심층 신경망 연구 재개
2000년대	- 데이터 기반 접근법으로 전환, 인터넷과 빅데이터 활용	- 2006: 딥러닝 개념 구체화 및 심층 신경망 가능성 제시 (힌튼)
2010년대	- 규칙 기반 시스템의 한계 인식	- 2012: 알렉스넷, 이미지넷 대회 우승, 딥러닝 성과 입증
		- 2014: 알파고 개발 (구글 딥마인드)
		- 2016: 알파고, 이세돌 9단 이기며 AI 성능 세계에 알림
2020년대	- GPT-3 등 대규모 언어 모델의 성과를 토대로 규칙 기반 접근 재조명	- GPT-3 및 GPT-4와 같은 대규모 언어 모델의 발전
	- 자연어 처리에서의 규칙 기반 모델의 보완적 사용	- 2021: AlphaFold로 단백질 구조 예측 혁신
	- 2023: GPT-4 및 대규모 언어 모델 지속 발전, 다양한 산업 분야 AI 기술 적용	

➤ 머신러닝은 무엇을 하려는 것인가?

- 문제해결 방법
 - 하나하나 지시하고 실행하는 방법
 - 새로운 문제 해결이 어렵다
 - 문제를 해결하는 일반적인 방법을 가르친다
 - 단순지시가 아님
 - 평가가 따름
 - 평가에 따라 다음 절차가 결정
 - 새로운 문제해결이 가능함



➤ 머신러닝이란

- 톰 미첼 Tom Mitchell 의 공학적 정의

“컴퓨터 프로그램이 어떤 작업 종류 T 에 속한 작업을 수행하면서 경험 E 에 따라서 P 로 측정하는 성능이 개선된다면, 이 프로그램은 T 와 성능 척도 P 에 대해 경험 E 로부터 **학습을 한다고** 말할 수 있다.”

예를 들어, 컴퓨터에 필기체를 인식하는 학습을 시킨다고 했을 때,

- ① **작업 T** : 필기체를 인식하고 분류하는 것
- ② **성능 P** : 필기체를 정확히 구분한 확률
- ③ **경험 E** : 필기체와 정확한 글자를 표시한 데이터 셋

머신러닝 연구 및 활용 과정

1. 문제 정의 (Problem Definition)

- 해결하고자 하는 문제를 명확히 정의한다. 예) 이미지 분류, 음성 인식, 추천 시스템 등

2. 데이터 수집 및 전처리 (Data Collection and Preprocessing)

- 문제 해결에 필요한 데이터 수집, 결측값 처리, 정규화, 특성 공학 등의 데이터 전처리 과정 수행.

3. 모델 선택 및 학습 (Model Selection and Training)

- 문제에 적합한 머신러닝 모델 선택, 수집한 데이터를 사용하여 모델 학습. 하이퍼파라미터 튜닝 포함.

4. 모델 평가 (Model Evaluation)

- 학습된 모델을 테스트 데이터나 교차검증을 통해 평가. 평가 지표로 모델 성능 측정.

5. 모델 개선 (Model Improvement)

- 모델 성능 향상을 위한 다양한 기법 적용. 예를 들어, 더 많은 데이터를 수집하거나, 더 복잡한 모델을 사용하거나, 앙상블 기법 적용 등.

6. 배포 및 운영 (Deployment and Operations)

- 최종 모델을 실제 환경에 배포, 운영. 모델의 실시간 성능 모니터링 및 주기적 재학습이 필요할 수도 있음.

7. 피드백 및 유지보수 (Feedback and Maintenance)

- 모델의 예측 성능을 지속적으로 모니터링. 필요에 따라 모델 업데이트. 새로운 데이터가 수집되면 주기적으로 모델을 재학습시켜 성능을 유지.

머신러닝 용어

1. 머신러닝 (Machine Learning)

- 컴퓨터가 명시적인 프로그램 없이도 데이터에서 학습하고 예측 또는 결정을 내리는 것을 가능하게 하는 인공지능의 하위 분야.

2. 지도학습 (Supervised Learning)

- 입력 데이터와 그에 대한 정답(라벨)이 주어진 상태에서 모델을 학습시키는 방법. 분류(Classification)와 회귀(Regression).

3. 비지도학습 (Unsupervised Learning)

- 입력 데이터만 주어지고 정답(라벨)이 없는 상태에서 데이터의 구조나 패턴을 발견하는 학습 방법. 군집화(Clustering)와 차원 축소(Dimensionality Reduction).

4. 강화학습 (Reinforcement Learning)

- 에이전트가 환경과 상호작용하며 보상을 최대화하는 행동을 학습하는 방법.

5. 특성 (Feature)

- 모델의 입력으로 사용되는 데이터의 개별 속성 또는 열(column).

6. 라벨 (Label)

- 지도학습에서 예측하려는 대상 변수. 예를 들어, 이메일이 스팸인지 아닌지를 예측하는 경우, "스팸" 또는 "비스팸"이 라벨이 됨.

7. 과적합 (Overfitting)

- 모델이 훈련 데이터에 너무 잘 맞춰져서 새로운 데이터에 일반화되지 않는 상태.

8. 과소적합 (Underfitting)

- 모델이 훈련 데이터의 패턴을 충분히 학습하지 못한 상태.

9. 교차검증 (Cross-Validation)

- 데이터를 여러 개의 부분으로 나누어 모델을 반복적으로 훈련하고 평가하는 방법. 예) K-겹 교차검증(K-Fold Cross-Validation)

10. 모델 파라미터 (Model Parameters)

- 모델 파라미터는 학습 과정에서 데이터로부터 학습되는 값들을 말함. 예) 선형 회귀 모델의 가중치(weight)와 절편(intercept)

11. 하이퍼파라미터 (Hyperparameter)

- 모델 학습 과정에서 사용자가 직접 설정해야 하는 매개변수. 예) 결정 트리의 깊이, 신경망의 학습률 등

12. 혼동 행렬 (Confusion Matrix)

- 분류 모델의 성능을 평가하기 위한 행렬. 행은 실제 클래스, 열은 예측 클래스를 나타내며, 네 가지 경우(True Positive, True Negative, False Positive, False Negative)를 나타냄.

머신러닝 관련 용어

13. 정밀도 (Precision)

- 모델이 양성으로 예측한 것 중 실제로 양성인 비율 $\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$

14. 재현율 (Recall)

- 실제 양성인 것 중에서 모델이 양성으로 정확히 예측한 비율. $\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$

15. F1 스코어 (F1 Score)

- 정밀도와 재현율의 조화평균을 계산한 값. $\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

16. 손실 함수 (Loss Function)

- 모델의 예측값과 실제값 간의 차이를 계산하는 함수. 모델 학습 과정은 손실함수 값을 최소화하는 것.

17. 경사 하강법 (Gradient Descent)

- 손실 함수를 최소화하기 위해 사용되는 최적화 알고리즘. 학습률(Learning Rate)로 모델 파라미터 업데이트.

18. 에포크 (Epoch)

- 전체 훈련 데이터셋을 한 번 모델에 통과시켜 학습하는 과정.

19. 배치 (Batch)

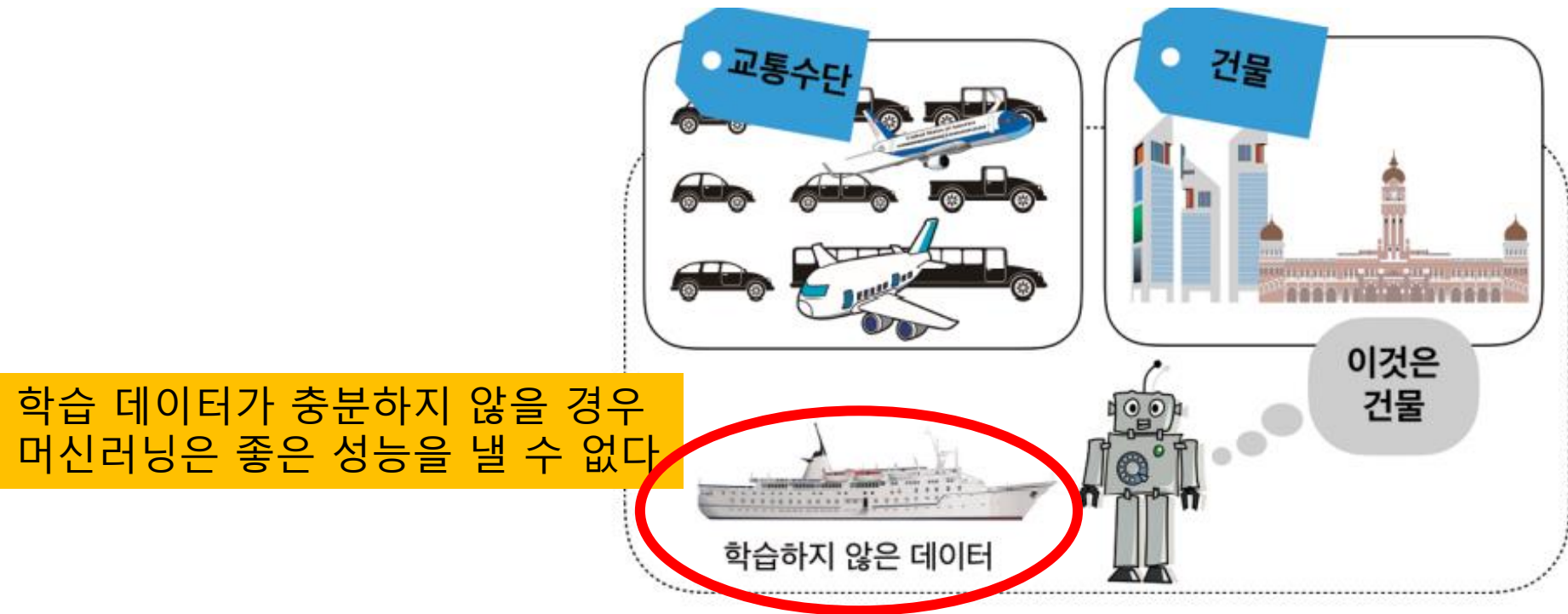
- 훈련 데이터셋을 나누어 모델에 전달하는 작은 묶음. 배치 단위로 모델이 업데이트됨.

20. 딥러닝 (Deep Learning)

- 다층 신경망을 사용하여 복잡한 패턴을 학습하는 머신러닝의 하위 분야. 일반적으로 인공 신경망(Artificial Neural Network, ANN)을 활용함.

결론적으로

- 머신러닝은 **파라미터**에 따라 동작하는 **알고리즘**algorithm을 선택하고, 이 알고리즘에 **데이터**를 제공하여 알고리즘이 더 나은 동작을 하도록 **파라미터를 수정**하는 것
- 따라서 머신러닝의 핵심적인 문제는 **알고리즘**과 **데이터**라고 할 수 있다.

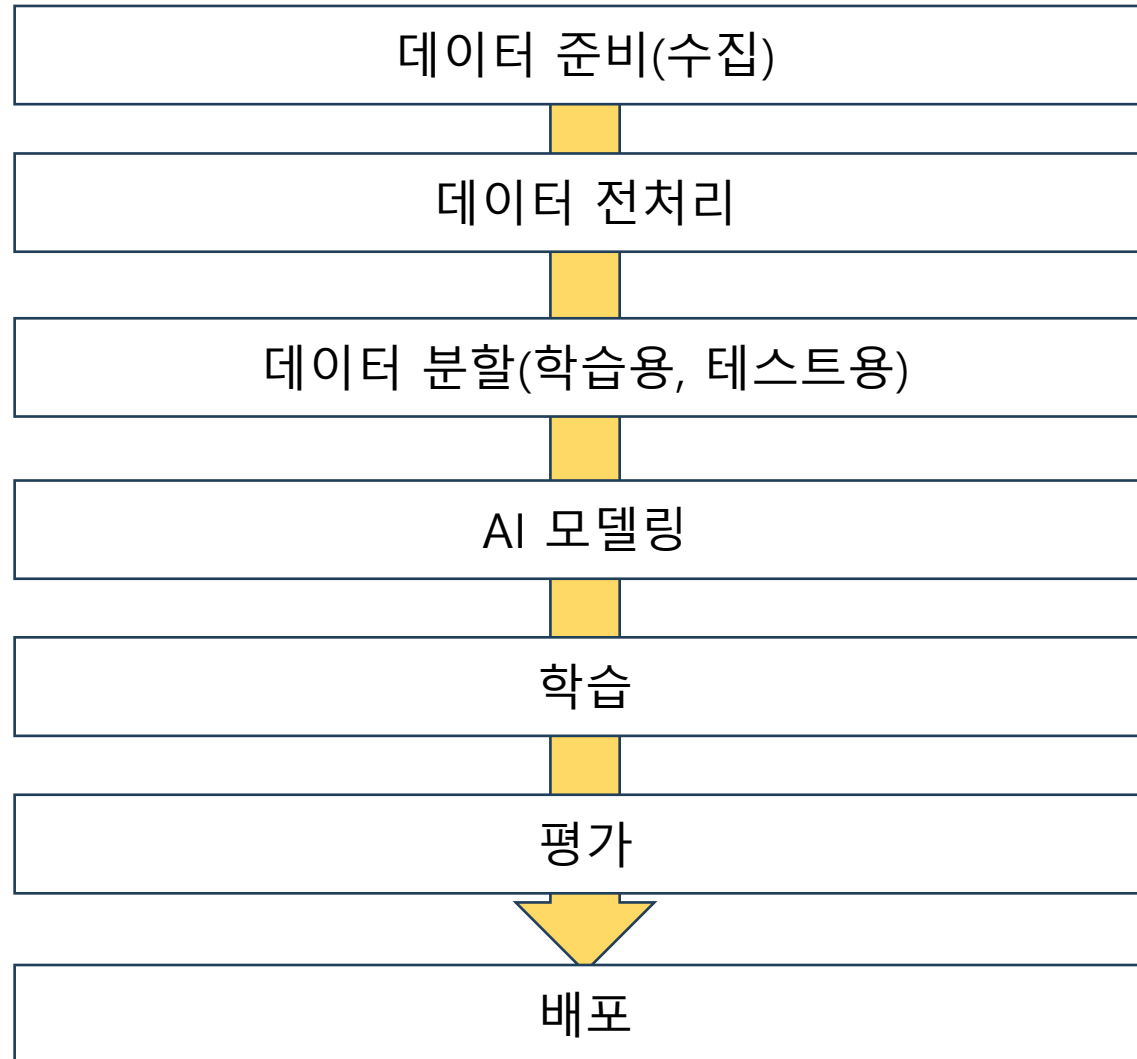


➤ 머신러닝과 데이터

- **데이터 편향** *data bias* - 확보된 데이터가 대표하는 모집단의 분포를 제대로 반영하지 못하고 일부의 특성만을 가지고 있는 경우
 - 편향의 원인은 두 가지
 - 너무 적은 수의 표본을 추출한 경우
 - 표집 방법이 잘못되어 모집단에 속한 대상을 골고루 추출하지 못 하는 경우.
- **부정확성** *inaccuracy* - 데이터의 품질이 낮아 많은 오류와 이상치, 잡음을 포함하고 있는 경우
- **무관함** *irrelavance* - 데이터는 많이 확보했지만, 이 데이터가 담고 있는 특성들이 학습하려고 하는 문제와는 무관한 데이터
- 머신러닝에서 데이터의 중요성은 아무리 강조해도 지나치지 않다

AI&머신러닝 체험

➤ AI 개발 과정



➤ AI 개발 체험: Teachable Machine

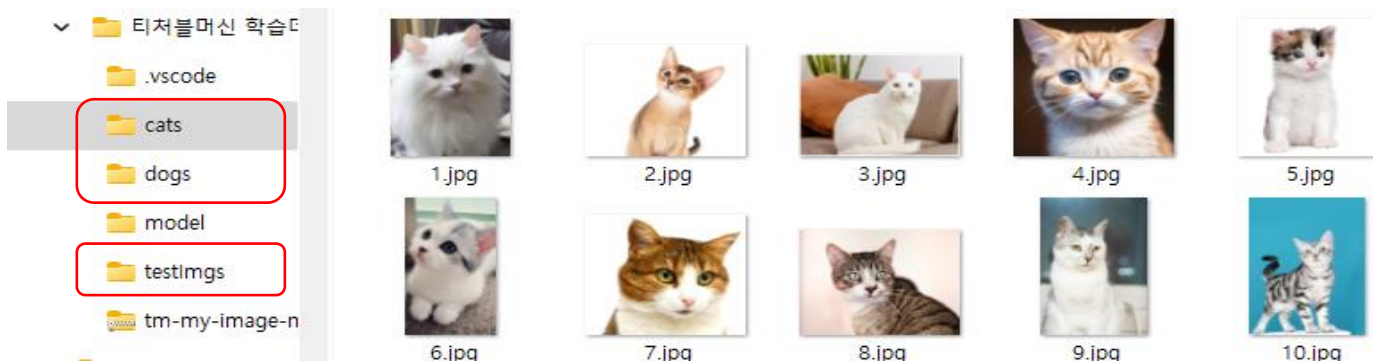
- 구글이 개발한 도구
- 교육 목적이나 프로토타이핑, 간단한 AI 프로젝트에 매우 유용하며, 머신러닝을 처음 접하는 사람들에게도 좋은 학습 도구
- 주요 특징
 1. 사용자 친화적 인터페이스: 복잡한 코드 없이 드래그 앤 드롭 방식으로 모델을 훈련시킬 수 있다.
 2. 다양한 입력 데이터 지원: 이미지, 소리, 포즈 데이터를 입력으로 사용할 수 있다.
 3. 빠른 학습 및 테스트: 실시간으로 데이터를 입력하고 모델을 테스트할 수 있다.
 4. 모델 내보내기: 훈련된 모델을 다운로드하거나 웹 프로젝트에 쉽게 통합할 수 있다.

➤ AI 개발 체험: Teachable Machine

➤ 강아지 고양이 분류 프로젝트 실습(tmProject 폴더)

모델 학습/테스트 데이터 준비

1. Dogs와 cats 폴더를 만들고, 관련 이미지를 10개 이상 수집하여 저장하기
2. Testimgs 폴더에는 AI 모델 테스트를 위한 개, 고양이 이미지 3개씩 수집하여 저장하기



3. 구글 teachable machine 접속, 이미지프로젝트 열기->표준이미지 모델 열기

새 프로젝트

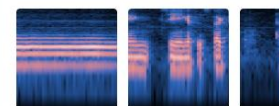
Drive에 있는 기존 프로젝트를 엽니다.

파일에서 기존 프로젝트를 엽니다.



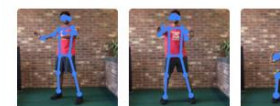
이미지 프로젝트

파일 또는 웹캠에서 가져온 이미지를 기반으로 학습시키세요.



오디오 프로젝트

파일 또는 마이크에서 가져온 1초 분량의 사운드를 기반으로 학습시키세요.



포즈 프로젝트

파일 또는 웹캠에서 가져온 이미지를 기반으로 학습시키세요.



새 이미지 프로젝트

표준 이미지 모델

대부분의 용도에 적합

224 x 224px 컬러 이미지

TensorFlow, TFLite, TF.js로 내보내기

모델 크기: 약 5mb

삽입된 이미지 모델

마이크로 컨트롤러에 적합

96 x 96px 그레이스케일 이미지

마이크로컨트롤러용 TFLite, TFLite, TF.js로 내보내기

모델 크기: 약 500kb

[이 모델을 지원하는 하드웨어를 확인하세요.](#)

➤ AI 개발 체험: Teachable Machine

➤ 강아지 고양이 분류 프로젝트 실습

4. cats와 dogs 클래스 만들고 1단계에서 수집한 이미지 업로드

The image shows a sequence of steps in the Teachable Machine interface for creating a 'dogs' class. On the left, a blue box with an upload icon and the text '업로드' (Upload) has a yellow arrow pointing to a 'dogs' class panel. This panel has a '파일' (File) tab selected, showing instructions to drag and drop images or use Google Drive. A grid of 11 dog images is visible. A second yellow arrow points to the right, where the 'dogs' class panel is shown with the '11 이미지 샘플' (11 image samples) tab selected, displaying the same 11 images. Below this, the 'cats' class panel is partially visible. On the far right, a '학습' (Train) button labeled '모델 학습시키기' (Train model) and a '고급' (Advanced) dropdown menu are shown.

dogs

파일

11 이미지 샘플

파일에서 이미지를 선택하거나 여기로 드래그 앤 드롭하세요.

Google Drive에서 이미지 가져오기

이미지가 정사각형 모양으로 잘립니다.

dogs

11 이미지 샘플

웹캠 업로드

cats

이미지 샘플 추가:

웹캠 업로드

학습

모델 학습시키기

고급

➤ AI 개발 체험: Teachable Machine

➤ 강아지 고양이 분류 프로젝트 실습

5. 모델 학습

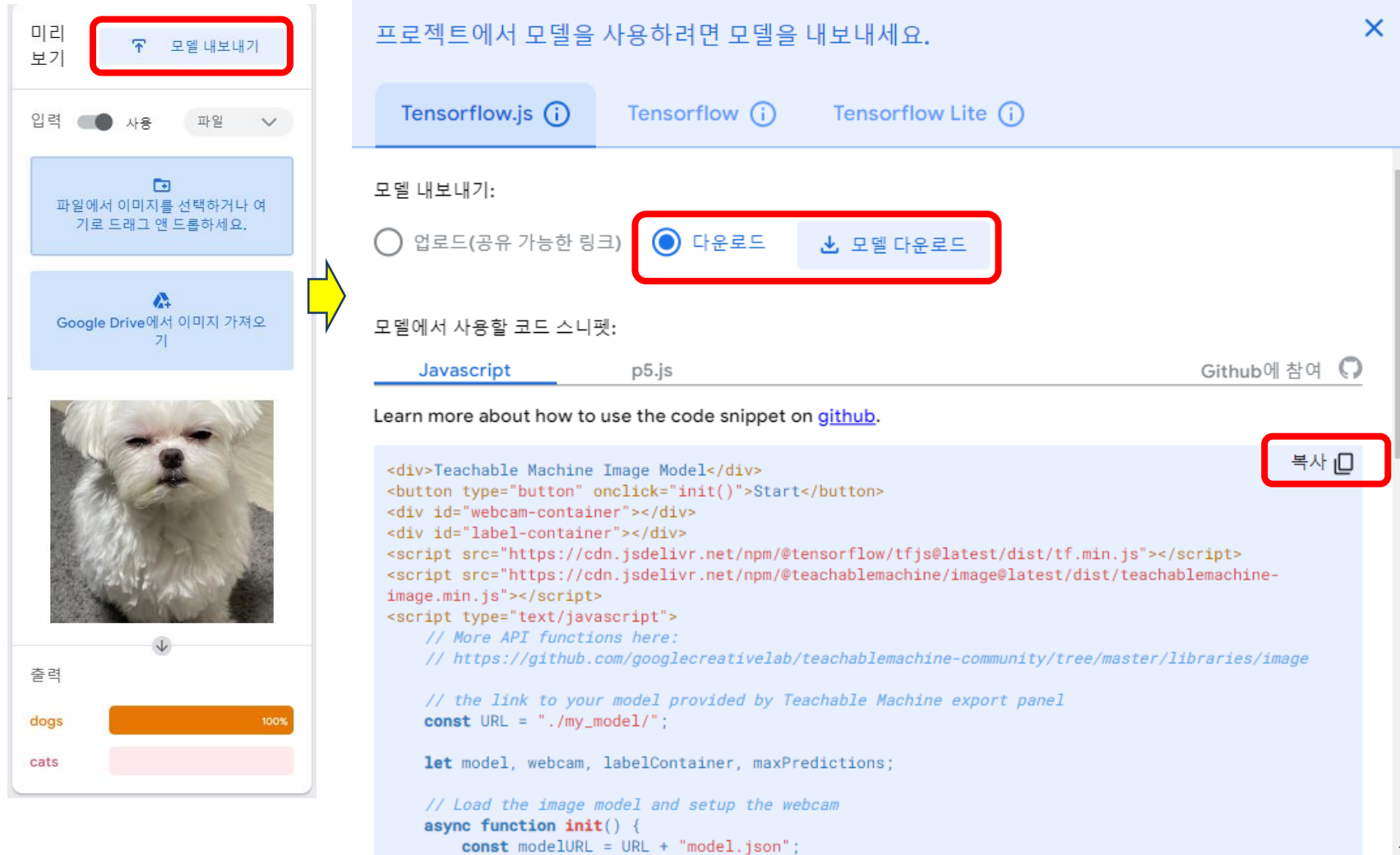
6. 모델 테스트

The screenshot displays the Teachable Machine web interface for training an image model. The browser address bar shows `teachablemachine.withgoogle.com/train/image`. The main interface is divided into several sections:

- Left Sidebar:** Contains a menu with "학습" (Train) and "고급" (Advanced). The "학습" button is highlighted with a red box and a yellow arrow pointing to the training area.
- Training Area:** Shows two classes: "dogs" and "cats". The "dogs" class has 11 image samples, and the "cats" class has 10 image samples. Below these is a dashed box labeled "클래스 추가" (Add Class).
- Right Panel:** Contains a "미리 보기" (Preview) section with a dropdown menu set to "파일" (File). Below this is a red box containing the text: "파일에서 이미지를 선택하거나 여기로 드래그 앤 드롭하세요." (Select an image from a file or drag and drop here). Further down is a "Google Drive에서 이미지 가져오기" (Import image from Google Drive) button. At the bottom right, there is a "출력" (Output) section showing a progress bar for "dogs" at 100% and a bar for "cats" that is currently empty.
- Center Bottom:** A red box highlights the "학습" (Train) button and the "고급" (Advanced) dropdown menu.

➤ AI 개발 체험: Teachable Machine

➤ 강아지 고양이 분류 프로젝트 실습 7. 모델 내보내기



미리 보기

입력 ☐ 사용 ☒ 파일

파일에서 이미지를 선택하거나 여기로 드래그 앤 드롭하세요.

Google Drive에서 이미지 가져오기

출력

dogs 100%

cats

프로젝트에서 모델을 사용하려면 모델을 내보내세요.

Tensorflow.js Tensorflow Tensorflow Lite

모델 내보내기:

☐ 업로드(공유 가능한 링크) ☒ 다운로드 [모델 다운로드](#)

모델에서 사용할 코드 스니펫:

Javascript p5.js Github에 참여

Learn more about how to use the code snippet on [github](#).

```
<div>Teachable Machine Image Model</div>
<button type="button" onclick="init()">Start</button>
<div id="webcam-container"></div>
<div id="label-container"></div>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablemachine-image.min.js"></script>
<script type="text/javascript">
  // More API functions here:
  // https://github.com/googlecreativelab/teachablemachine-community/tree/master/libraries/image

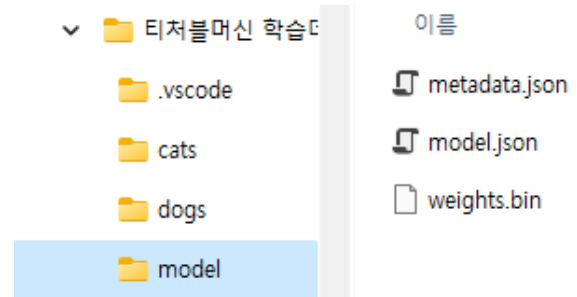
  // the link to your model provided by Teachable Machine export panel
  const URL = "/my_model/";

  let model, webcam, labelContainer, maxPredictions;

  // Load the image model and setup the webcam
  async function init() {
    const modelURL = URL + "model.json";
```

복사

8. 다운로드 받은 압축파일 풀기
폴더명: model



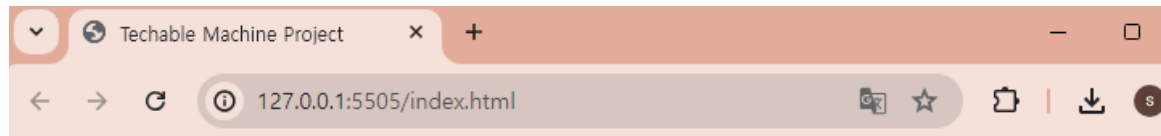
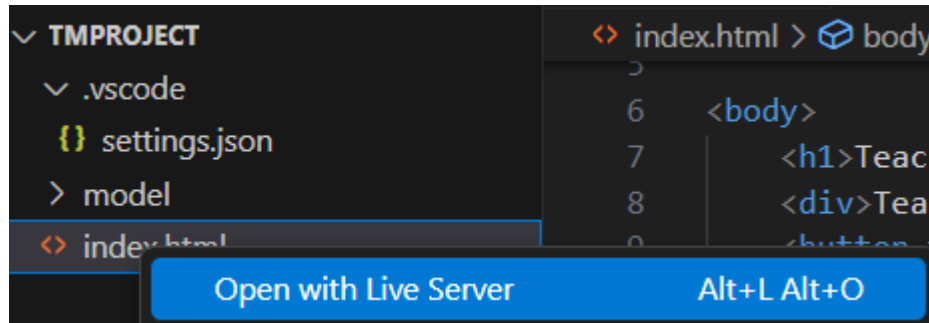
9. 코드복사 후
index.html에 붙여넣기

➤ AI 개발 체험: Teachable Machine

➤ 강아지 고양이 분류 프로젝트 실습

10. 프로젝트 실행하기

- vscode 에서 tmProject 폴더 열기
- Live Server로 index.html 파일 열기



Teachable Machine Dog and Cat Classification

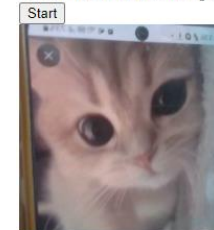
Teachable Machine Image Model

Start



Teachable Machine D

Teachable Machine Image Model



dogs: 0.02
cats: 0.98

머신러닝 종류

지도학습


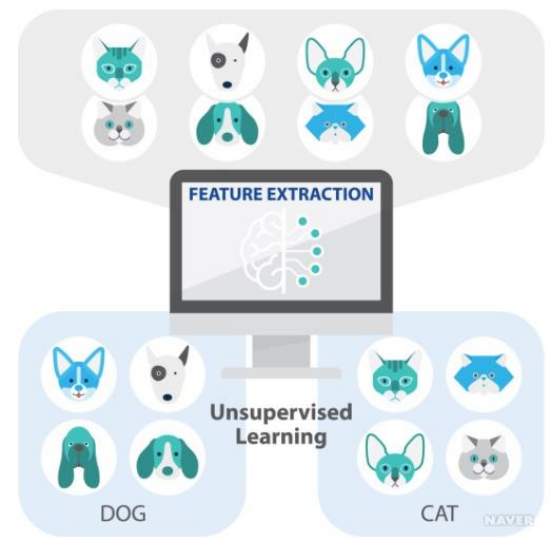
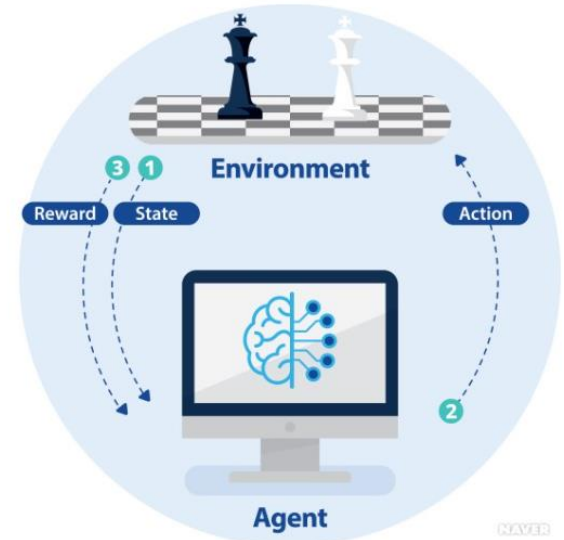
비지도학습

강화학습

머신러닝 종류

- 컴퓨터가 명시적으로 프로그램되지 않아도 데이터를 통해 학습하여 특정 작업을 수행할 수 있도록 하는 기술

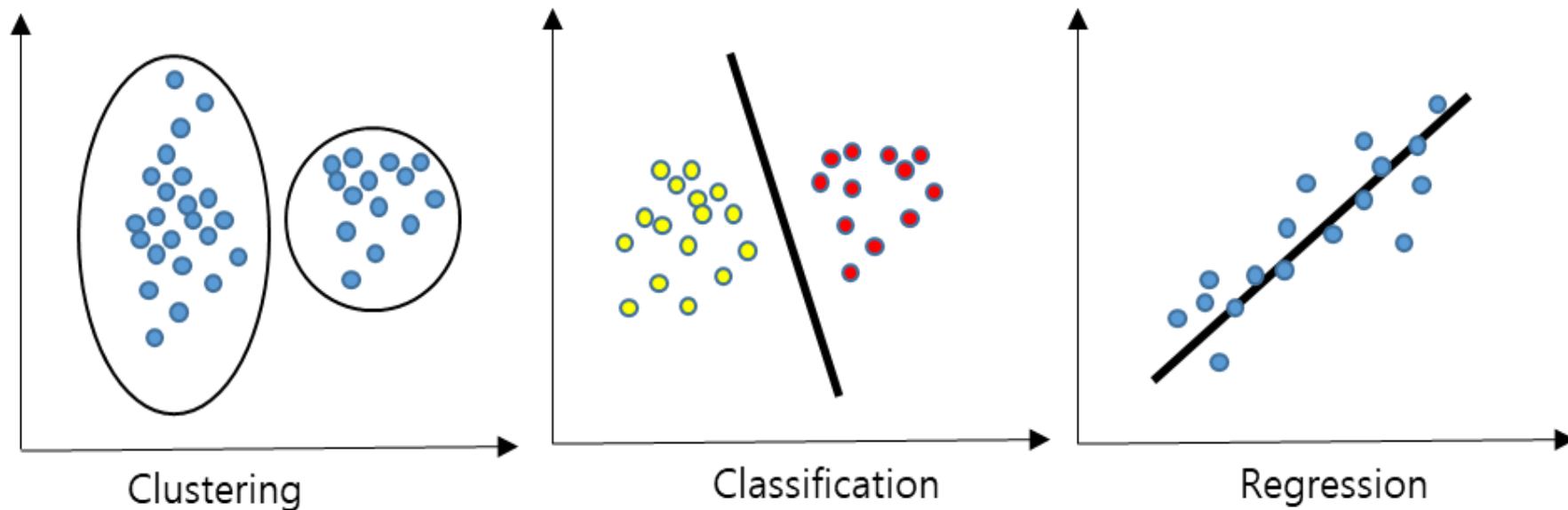
➤ 학습방식에 따른 구분

지도학습	비지도학습	강화 학습
이미 정답이 있는 데이터를 통해 모델을 학습시키는 방법	정답이 없는 데이터를 통해 패턴을 찾는 방법	행동과 보상을 통해 학습하는 방법
		

● 머신러닝 종류

➤ 해결해야 하는 문제에 따른 구분

1. 분류 문제(Classification): 범주형 레이블 예측. 예를 들어 예(Y)/아니오(N) 등
2. 회귀 문제(Regression): 연속된 값 예측.
3. 군집화 문제(Clustering): 비슷한 특징을 가진 것끼리 묶음



• 지도학습

- 회귀(Regression)
 - 선형(linear)과 비선형(nonlinear) 회귀
 - 경사하강법(Gradient Descent)
 - 편향(Bias)과 분산(Variance) 트레이드 오프 (Trade-off)
- 분류
 - 로지스틱(Logistic)과 소프트맥스(Softmax) Regression: 함수 활용한 분류 모델
 - 서포트 벡터 머신(Support Vector Machine: SVM)
 - 결정트리(Decision Tree)
 - 선형판별분석(Linear Discriminant Anyalysis: LDA)
- 앙상블 학습: 모델 여러 개를 학습하여 예측(추론)한 결과를 평균하여 선택
 - Bagging
 - Boosting

• 비지도 학습

- 군집화
 - K-Means
 - Mean Shift
 - Gaussian Mixture Model
 - DBSCAN
- 차원 축소(전처리에 사용됨)
 - Principal Component Analysis: PCA)
 - Singular Value Decomposition(SVD)

첫 번째 머신러닝 프로그램

k-NN 분류 알고리즘

지도학습

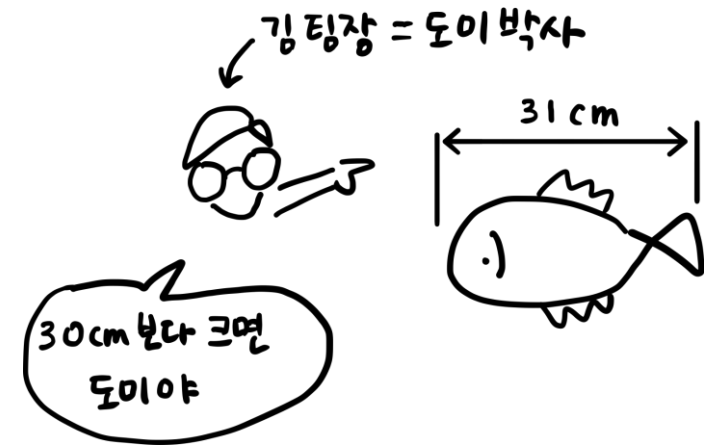
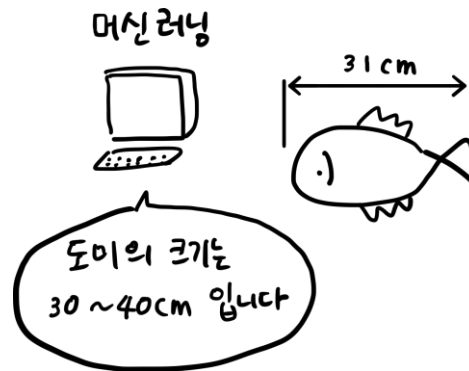
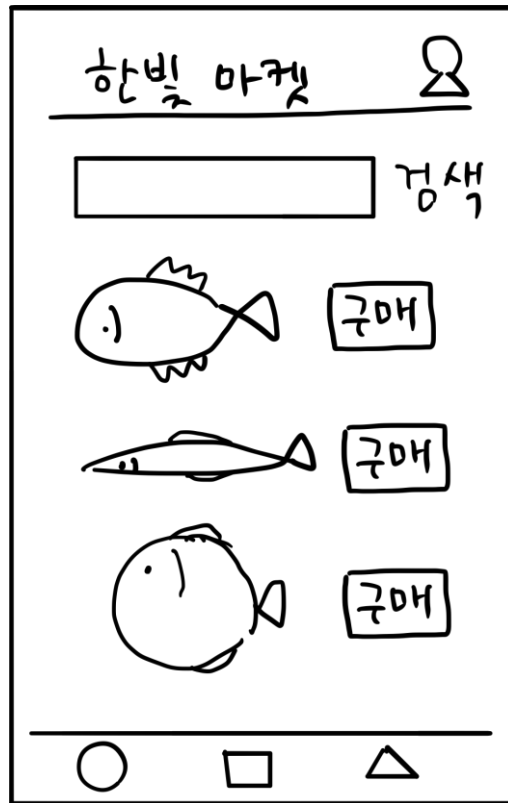
K-Nearest Neighborhood

학습데이터 편향성 문제

학습데이터 분할 문제

• k-NN 알고리즘

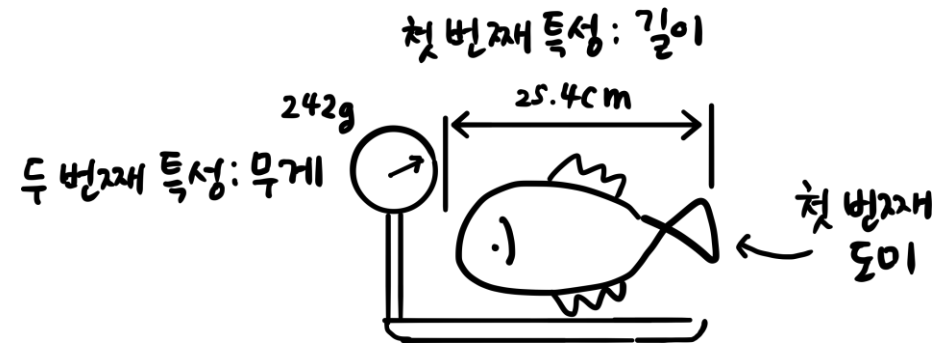
- 생선 분류 문제
- 도미 vs 빙어 이진 분류(binary classification) 문제



```
if fish_length >= 30:  
    print("도미")
```

• k-NN 알고리즘

- 데이터 준비: 도미 무게와 길이 측정 데이터



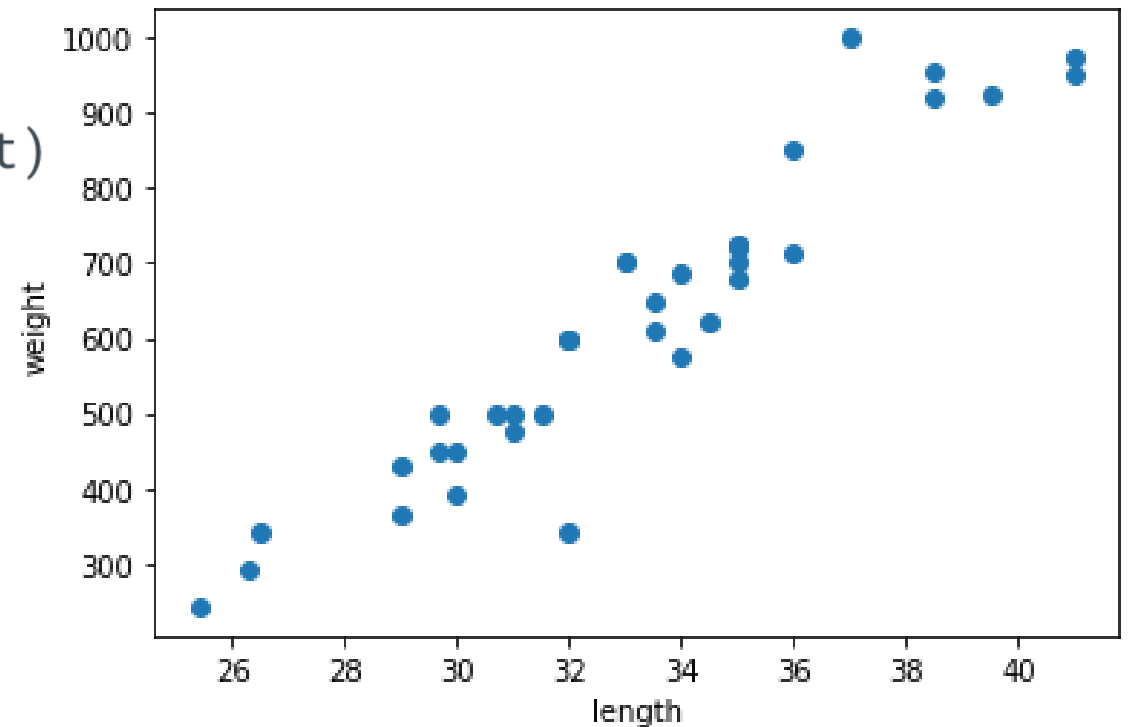
```
bream_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7,
                 31.0, 31.0, 31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5,
                 34.0, 34.0, 34.5, 35.0, 35.0, 35.0, 35.0, 36.0, 36.0, 37.0,
                 38.5, 38.5, 39.5, 41.0, 41.0]
bream_weight = [242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0,
                450.0, 500.0, 475.0, 500.0, 500.0, 340.0, 600.0, 600.0,
                700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,
                700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0,
                925.0, 975.0, 950.0]
```

• k-NN 알고리즘

- 길이와 무게 특성 상관 관계 분석: 산점도(scatter plot)

```
import matplotlib.pyplot as plt

plt.scatter(bream_length, bream_weight)
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```

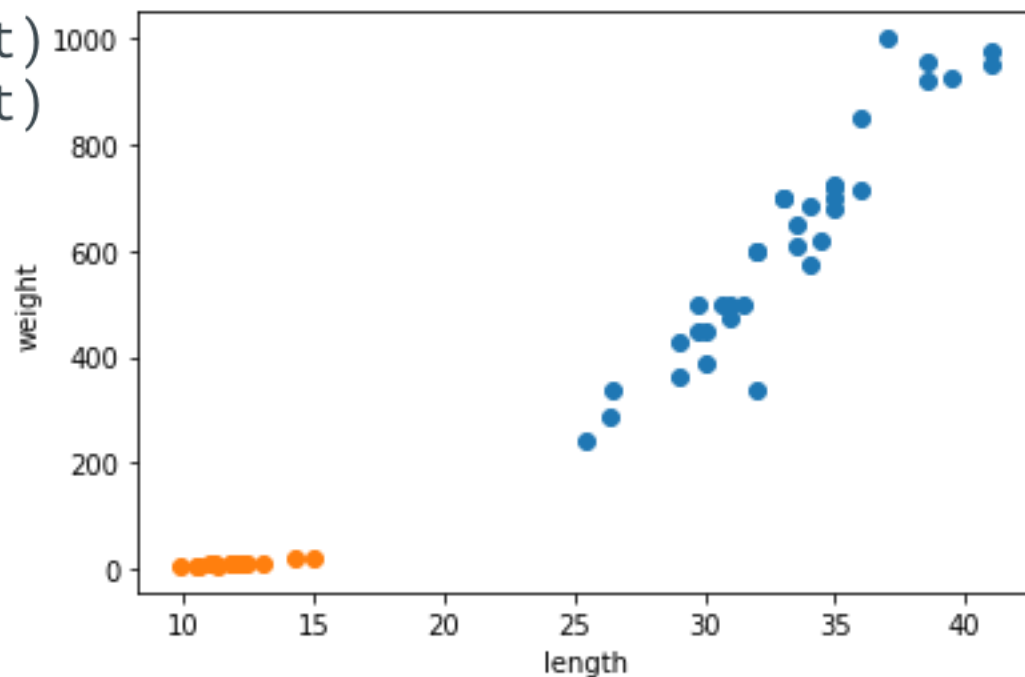


• k-NN 알고리즘

- 데이터 준비: 빙어 무게와 길이 측정 데이터

```
smelt_length = [9.8, 10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2,  
               12.4, 13.0, 14.3, 15.0]  
smelt_weight = [6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4,  
               12.2, 19.7, 19.9]
```

```
plt.scatter(bream_length, bream_weight)  
plt.scatter(smelt_length, smelt_weight)  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



• k-NN 알고리즘

- 도미와 빙어 데이터 합치기(같은 특성끼리)

```
length = bream_length+smelt_length  
weight = bream_weight+smelt_weight
```

도미 35개의 길이 빙어 14개의 길이

length = [25.4, 26.3, ... , 41.0, 9.8, ... , 15.0]

도미 35개의 무게 빙어 14개의 무게

weight = [242.0, 290.0, ... , 950.0, 6.7, ... , 19.9]



사이킷런이 기대하는 데이터 형태

49개의 생선

길이	무게
[[25.4, 242.0],	
[26.3, 290.0],	
.	.
.	.
.	.
[15.0, 19.9]]	

• k-NN 알고리즘

- 사이킷런 자료구조로 만들기: 특성 묶기

```
fish_data = [[l, w] for l, w in zip(length, weight)]
```

```
[[25.4, 242.0], [26.3, 290.0], [26.5, 340.0], [29.0, 363.0], [29.0, 430.0],  
[29.7, 450.0], [29.7, 500.0], [30.0, 390.0], [30.0, 450.0], [30.7, 500.0],  
[31.0, 475.0], [31.0, 500.0], [31.5, 500.0], [32.0, 340.0], [32.0, 600.0],  
[32.0, 600.0], [33.0, 700.0], [33.0, 700.0], [33.5, 610.0], [33.5, 650.0],  
[34.0, 575.0], [34.0, 685.0], [34.5, 620.0], [35.0, 680.0], [35.0, 700.0],  
[35.0, 725.0], [35.0, 720.0], [36.0, 714.0], [36.0, 850.0], [37.0, 1000.0],  
[38.5, 920.0], [38.5, 955.0], [39.5, 925.0], [41.0, 975.0], [41.0, 950.0],  
[9.8, 6.7], [10.5, 7.5], [10.6, 7.0], [11.0, 9.7], [11.2, 9.8], [11.3, 8.7],  
[11.8, 10.0], [11.8, 9.9], [12.0, 9.8], [12.2, 12.2], [12.4, 13.4],  
[13.0, 12.2], [14.3, 19.7], [15.0, 19.9]]
```

• k-NN 알고리즘

- 정답(label) 준비

```
fish_target = [1]*35 + [0]*14
```

[1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

• k-NN 알고리즘

■ k-최근접 이웃

```
from sklearn.neighbors import KNeighborsClassifier
```

```
kn = KNeighborsClassifier()
```

```
kn.fit(fish_data, fish_target) }
```

 KNN 모델 훈련

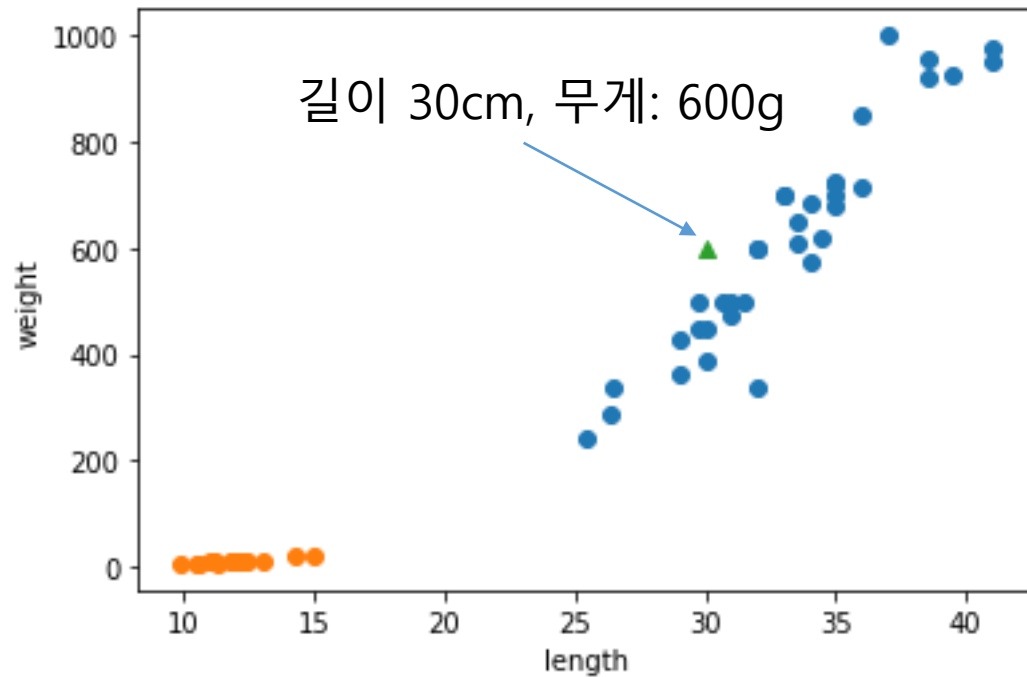
```
kn.score(fish_data, fish_target) }
```

 KNN 모델 평가

1.0

• k-NN 알고리즘

■ 새로운 생선 예측



```
kn.predict([[30, 600]])
```

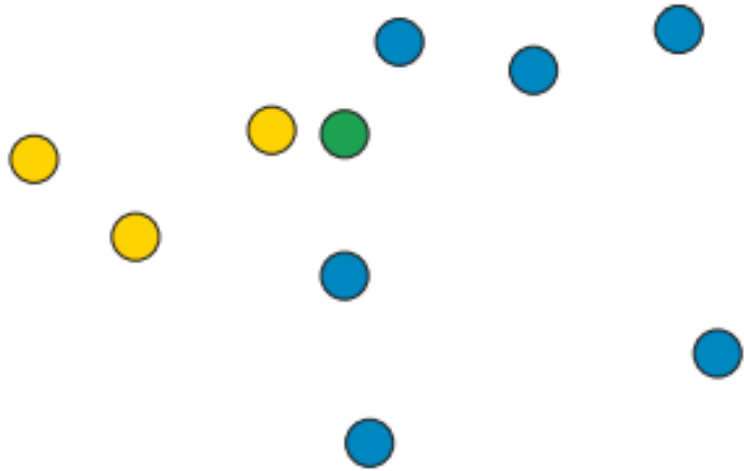
```
array([1])
```

• K-최근접 이웃(K-Nearest Neighbor) 알고리즘

- 새로운 데이터가 들어왔을 때 특성 공간 내에 데이터 간의 거리가 가까운 데이터를 찾아서 그것의 레이블의 값으로 분류하는 알고리즘
- 이때 K는 특성값을 기준으로 가장 거리가 가까운 데이터의 개수를 의미

예) 초록색으로 표시된 새로운 데이터를 노란색으로 분류할지 파란색으로 분류할지를 결정하는 문제

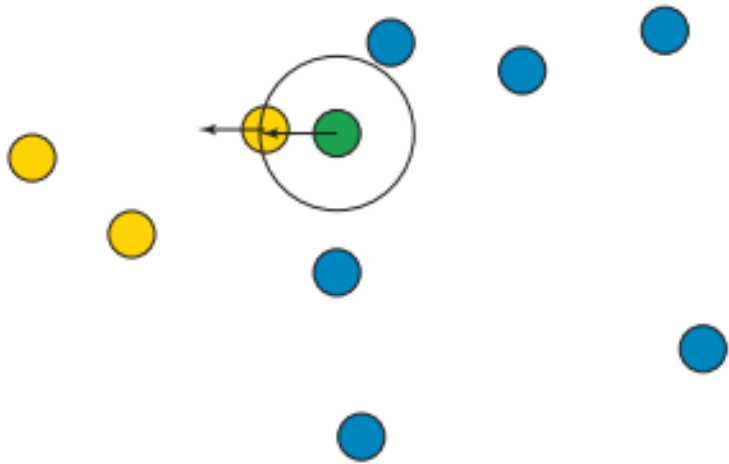
데이터	레이블
●	노란색
●	파란색
●	(모름.)



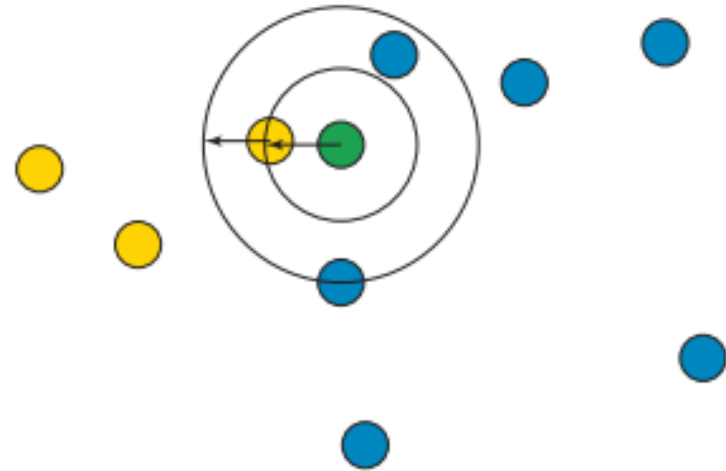
▲ 노란색, 파란색 색깔 분류 문제

• K-NN 알고리즘

K=1인 경우



K=3인 경우



• K-NN 알고리즘

[K-NN 이용한 분류기 정의]

- ❶ 새로운 샘플 데이터와 분류된 데이터 사이의 거리 계산
- ❷ 새로운 샘플 데이터와 가까운 거리 순서로 정렬
- ❸ 가장 가까운 K개 데이터 중에서 레이블별 빈도 세기
- ❹ 최다 빈도의 레이블값 출력하기

• k-NN 알고리즘: 무조건 도미?

- 데이터 개수가 49개일 때, kNN의 k값을 49로 지정할 경우

```
kn49 = KNeighborsClassifier(n_neighbors=49)
```

```
kn49.fit(fish_data, fish_target)  
kn49.score(fish_data, fish_target)
```

```
0.7142857142857143
```

```
print(35/49)
```

```
0.7142857142857143
```

• 데이터 편향성

K-NN 알고리즘에서 k 의 값은 가장 가까이 이웃한 데이터를 고려하여 분류 결정된다.

샘플 데이터 개수가 고르지 않으면 특정 클래스에 편향되어 결과가 만들어지고, 이런 경우, K 값이 클수록 데이터 편향성이 강하게 나타난다.

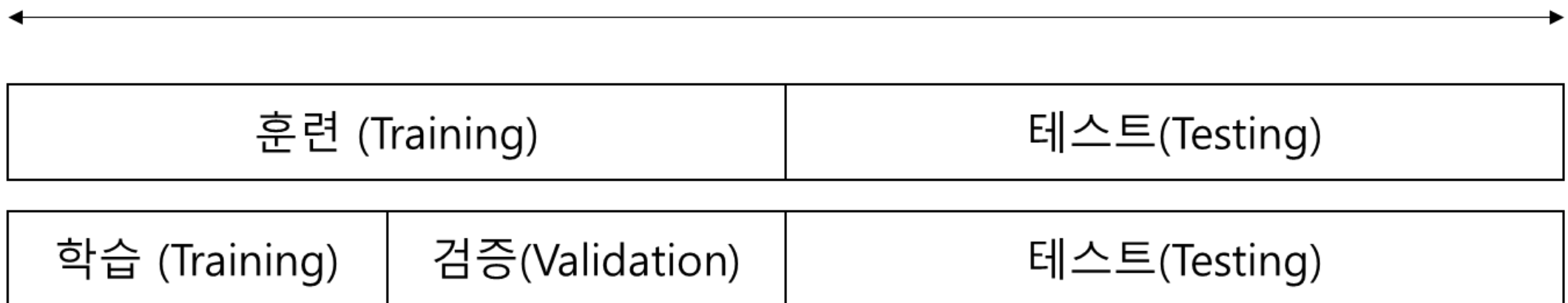
• 학습용 데이터를 분할하는 이유

머신러닝은 작업의 규칙을 찾기 위해 샘플 데이터로부터 학습한다. 학습 결과 만들어진 머신러닝 모델을 학습에 참여한 샘플데이터로 평가한다면 항상 좋은 성과를 낼 수 있다. 따라서 모델 평가를 위해서는 학습에 참여하지 않은 별도의 샘플 데이터가 있어야 한다.

학습에 참여한 데이터를 훈련데이터, 평가를 위한 데이터를 테스트 데이터라고 한다.

즉, 머신러닝을 위한 전체 데이터를 학습을 위한 훈련용 데이터, 평가를 위한 테스트용 데이터로 분할하여 구분하거나 또는 훈련용, 검증용, 테스트용으로 분리하기도 한다.

전체 데이터



• 훈련 세트와 테스트 세트

49개 샘플

2개의 특성

$\left\{ \begin{array}{l} [[25.4, 242.0], \\ [26.3, 290.0], \\ \vdots \\ [15.0, 19.9]] \end{array} \right\}$

훈련 세트 35개

테스트 세트 14개

```
train_input = fish_data[:35]  
train_target = fish_target[:35]
```

```
test_input = fish_data[35:]  
test_target = fish_target[35:]
```

- 테스트 세트에서 평가하기

```
from sklearn.neighbors import KNeighborsClassifier
```

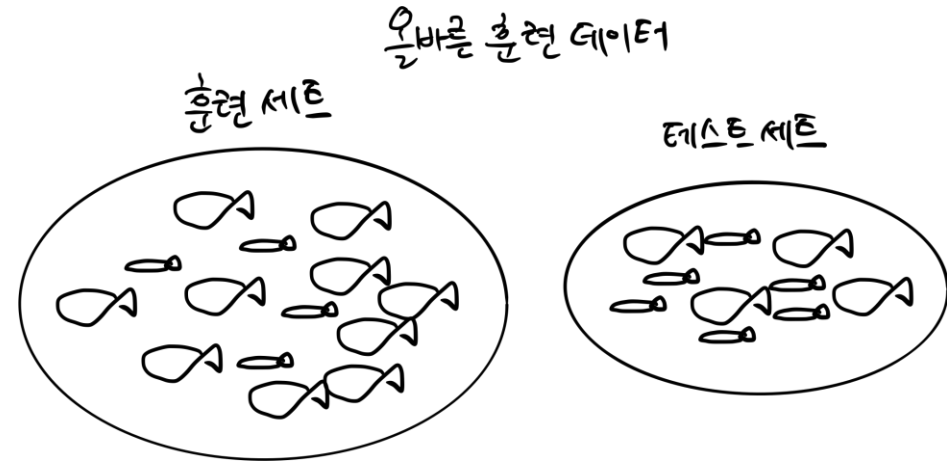
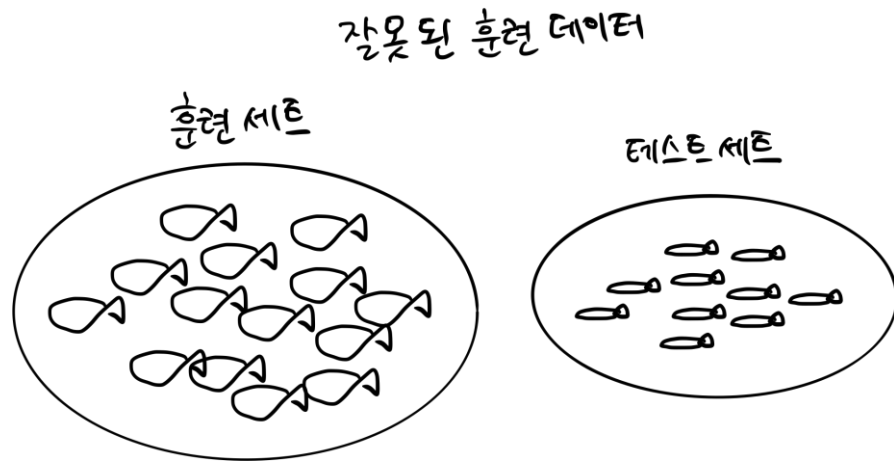
```
kn = KNeighborsClassifier()
```

```
kn = kn.fit(train_input, train_target)
```

```
kn.score(test_input, test_target)
```

```
0.0
```

• 샘플링 편향



• 넘파이 사용하기



numpy.org

1차원 배열

→ 축 1

3	10	2	5	7	8	9
---	----	---	---	---	---	---

2차원 배열

→ 축 2

축 1 ↓	3	10	2	...			

3차원 배열

축 3 ↗	축 2 →	축 1 ↓	3	10	2	...				

```
import numpy as np
```

```
input_arr = np.array(fish_data)
```

```
target_arr = np.array(fish_target)
```

```
print(input_arr)
```

2개의 열 (특성)

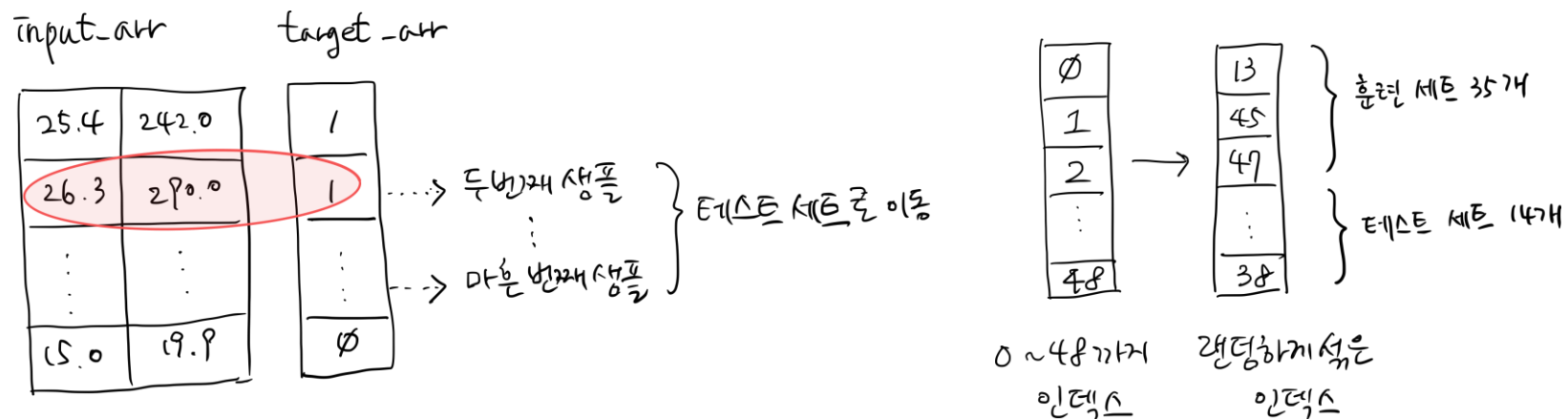
49개 행 (샘플)

{

- [25.4, 242.0],
- [26.3, 290.0],
- ⋮
- [15.0, 19.9]

}

• 데이터 섞기



```
np.random.seed(42)
```

```
index = np.arange(49)  
np.random.shuffle(index)
```

```
[13 45 47 44 17 27 26 25 31 19 12  4 34  8  3  6 40 41 46 15  9 16 24 33  
30  0 43 32  5 29 11 36  1 21  2 37 35 23 39 10 22 18 48 20  7 42 14 28  
38]
```

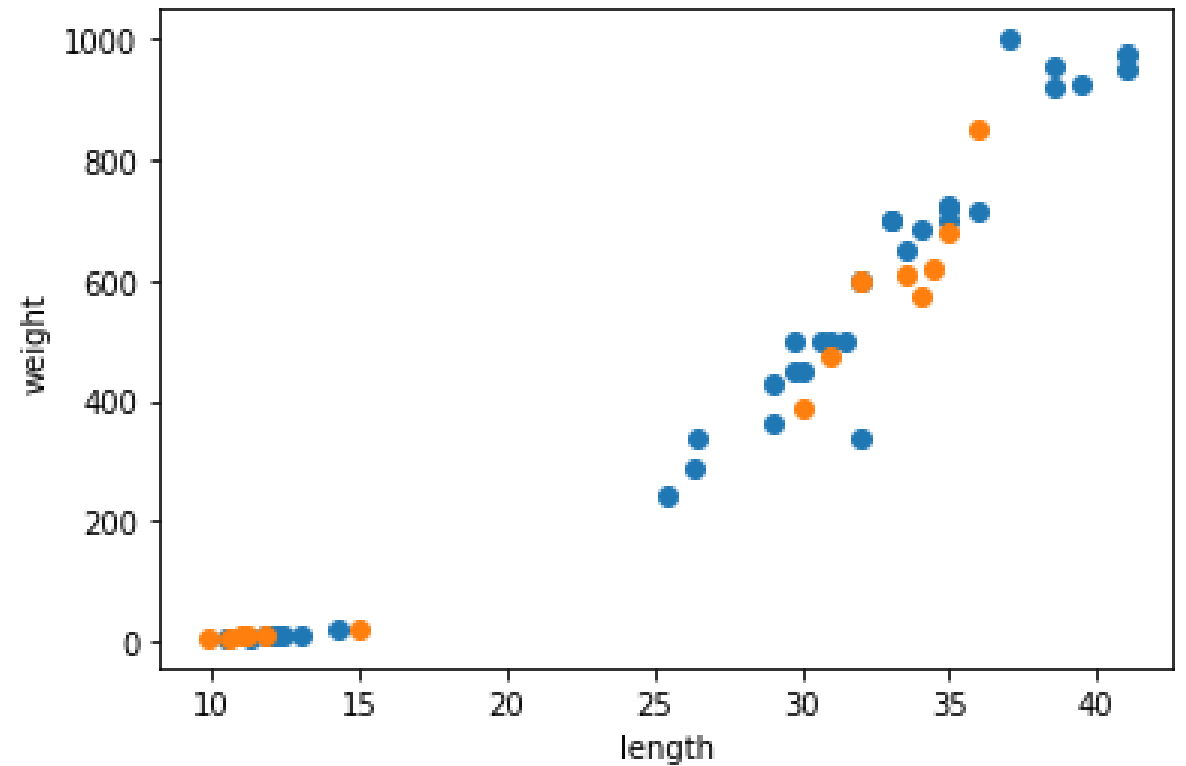
```
train_input = input_arr[index[:35]]  
train_target = target_arr[index[:35]]
```

• 데이터 나누고 확인하기

```
test_input = input_arr[index[35:]]
test_target = target_arr[index[35:]]

import matplotlib.pyplot as plt

plt.scatter(train_input[:, 0], train_input[:, 1])
plt.scatter(test_input[:, 0], test_input[:, 1])
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



• k-NN 알고리즘 평가

```
kn = kn.fit(train_input, train_target)
```

```
kn.score(test_input, test_target)
```

1.0

✓ 새로운 데이터 분류에 k-NN 모델을 적용해도 괜찮은가?

