pyeongchon tech

Supervised Learning 1

지도 학습 맛보기 1

소소오빠 황혁진

개발 환경 설정

생선 분류 모델 만들기

훈련 세트 vs 테스트 세트

데이터 전처리

개발 환경 설정

생선 분류 모델 만들기

훈련 세트 vs 테스트 세트

데이터 전처리

개발 환경 설정

생선 분류 모델 만들기

훈련 세트 vs 테스트 세트

데이터 전처리

개발 환경 설정

- 코랩 : 구글 계정이 있으면 누구나 사용할 수 있는 웹 브라우저 기반의 파이썬 코드 실행 환경
- 노트북 : 코랩의 프로개름 작성 단위이며 일반 프로그램 파일과 달리 대화식으로 프로그램을 만들 수 있기 때문에 데이터 분석이나 교육에 매우 적합하다. 노트북에는 코드, 코드의 실행 결과, 문서를 모두 저장하여 보관할 수 있다.
- 구글 드라이브 : 구글이 제공하는 클라우드 파일 저장 서비스이다. 코랩에서 만든 노트북은 자동으로 구글 드라이브의 'Colab Notebooks' 폴더에 저장되고 필요할때 다시 코랩에서 열 수 있다.

코랩(Colaboratory)

- 클라우드 기반의 주피터 노트북 개발 환경
- 누구나 동일한 결과를 표현할 수 있게 실습 환경을 제공한다.
- 웹 브라우저에서 무료로 파이썬 프로그램 테스트, 저장 가능하게 하는 서비스
- 심지어 머신러닝 프로그램도 만들 수 있다.
- 머신러닝은 컴퓨터 성능과 상관 없이 프로그램을 실습해볼 수 있다.
- 실습 주소 (https://colab.research.google.com)

- 구성이 필요하지 않음
- 무료로 GPU 사용
- 간편한 공유

학생이든, **데이터 과학자**든, **AI 연구원**이든 Colab으로 업무를 더욱 간편하게 처리할 수 있습니다. Colab 소개 영상에서 자세한 내용을 확인하거나 아래에서 시작해 보세요.

+ 코드 + 텍스트

시작하기

지금 읽고 계신 문서는 정적 웹페이지가 아니라 코드를 작성하고 실행할 수 있는 대화형 환경인 **Colab 메모장**입니다.

예를 들어 다음은 값을 계산하여 변수로 저장하고 결과를 출력하는 간단한 Python 스크립트가 포함된 **코드 셀**입니다.

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

```
86400
```

위 셀의 코드를 실행하려면 셀을 클릭하여 선택한 후 코드 왼쪽의 실행 버튼을 누르거나 단축키 'Command/Ctrl+Enter'를 사용하세요. 셀을 클릭하면 코드 수정을 바로 시작할 수 있습니다.

특정 셀에서 정의한 변수를 나중에 다른 셀에서 사용할 수 있습니다.

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week
```

```
604800
```

Colab 메모장을 사용하면 **실행 코드**와 **서식 있는 텍스트**를 **이미지**, **HTML**, **LaTeX** 등과 함께 하나의 문서로 통합할 수 있습니다. Colab 메모장을 만들면 Google Drive 계정에 저장됩니다. Colab 메모장을 간편하게 공유하여 동료나 친구들이 댓글을 달거나 수정하도록 할 수 있습니다. 자세히 알아보려면 Colab 개요를 참조하세요. 새 Colab 메모장을 만들려면 위의 파일 메뉴를 사용하거나 다음 링크로 이동하세요. 새 Colab 메모장 만들기

Colab 메모장은 Colab에서 호스팅하는 Jupyter 메모장입니다. Jupyter 프로젝트에 관해 자세히 알아보려면 jupyter.org를 참조하세요.

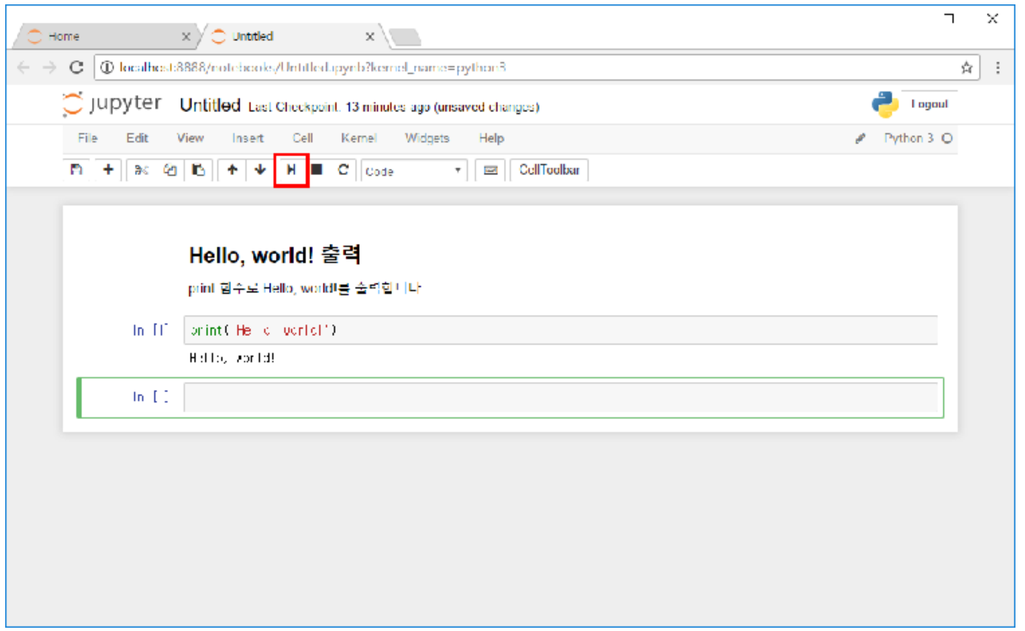
데이터 과학

Colab을 통해 인기 있는 Python 라이브러리를 최대한 활용하여 데이터를 분석하고 시각화할 수 있습니다. 아래 코드 셀에서는 **Numpy**를 사용하여 임의의 데이터를 생성하고 **매트플롯립**으로 이를 시각화합니다. 셀을 클릭하면 코드 수정을 바로 시작할 수 있습니다.

```
[ ] import numpy as np
```

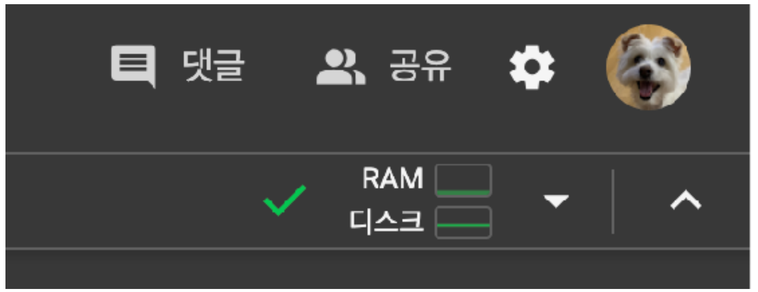
노트북

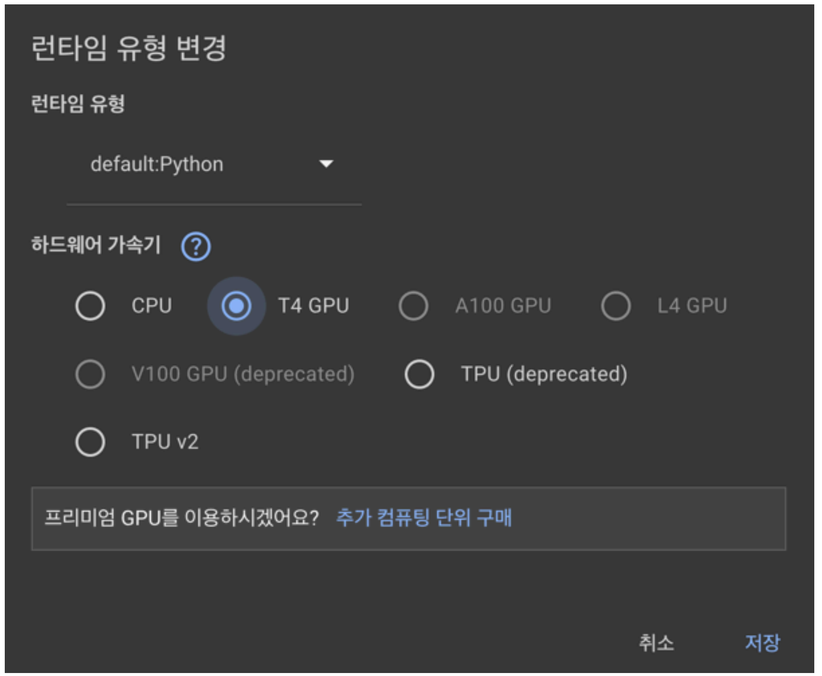
- 코랩은 구글이 대화식 프로그래밍 환경인 주피터 (Jupyter) 노트북을 커스터마이징 한 것이다.



노트북

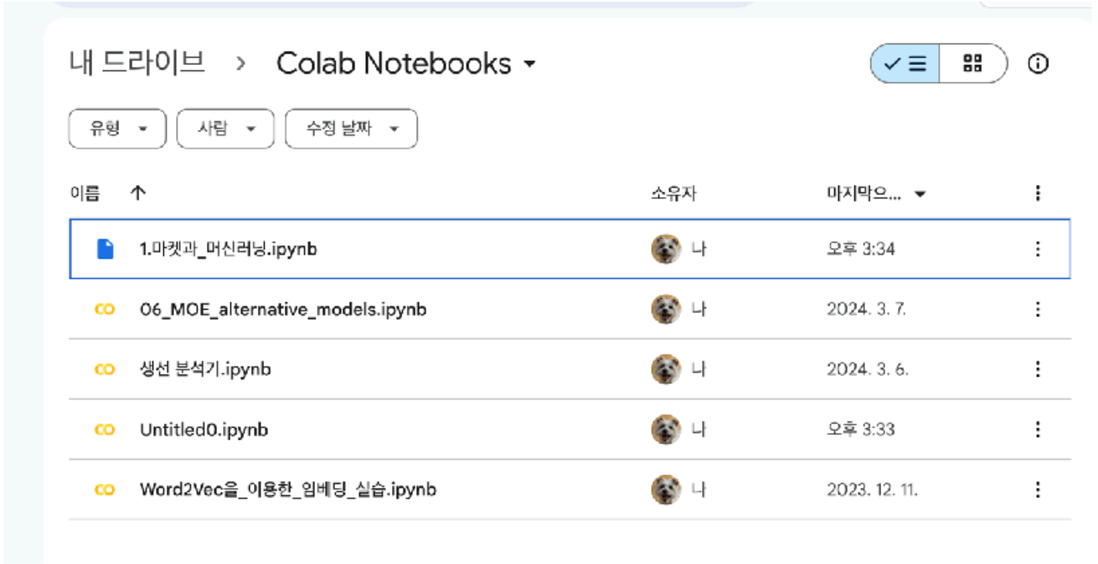
- 노트북을 생성하고 연결 버튼을 누르면 구글 클라우드 가상 서버(Virtual Machine)을 사용하게 된다.
- 할당된 RAM, 디스크를 볼 수 있으며 심지어 GPU도 한정적으로 사용할 수 있다.





구글드라이브

- 노트북을 저장하면 구글 드라이브에 "Colab Notebooks"에 자동으로 저장되어 관리된다.



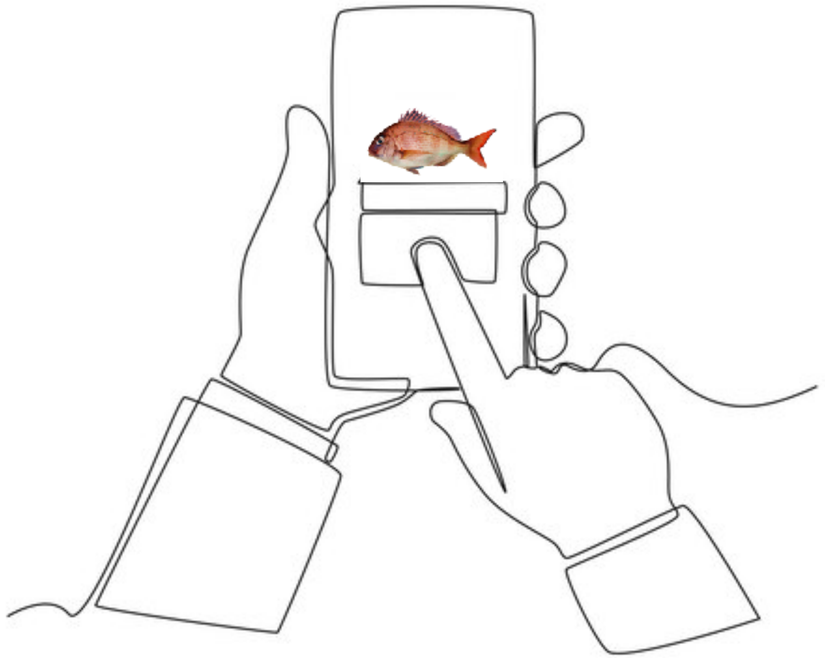
개발 환경 설정

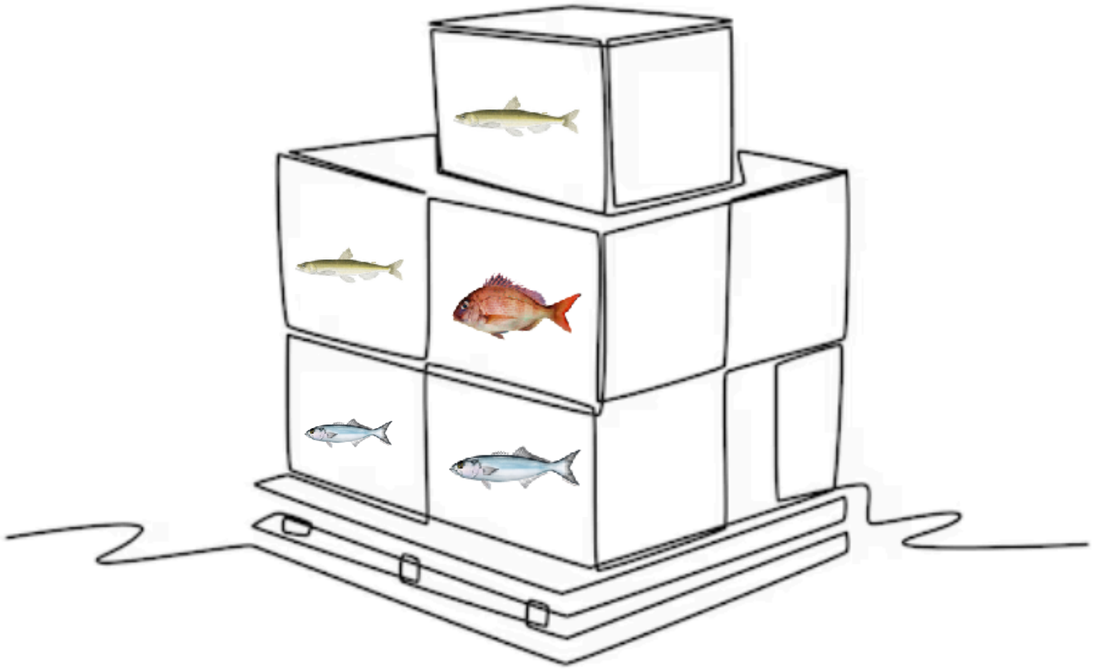
생선 분류 모델 만들기

훈련 세트 vs 테스트 세트

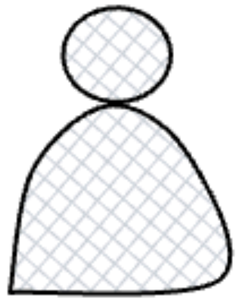
데이터 전처리

생선 분류





생선 분류



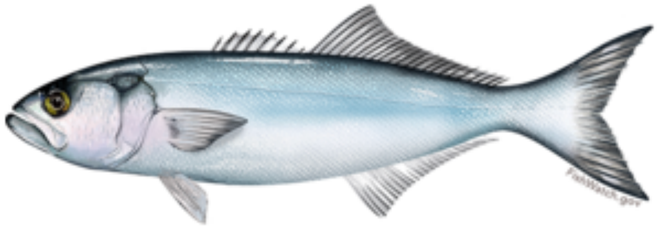
분류 담당

?

생선 분류

도미

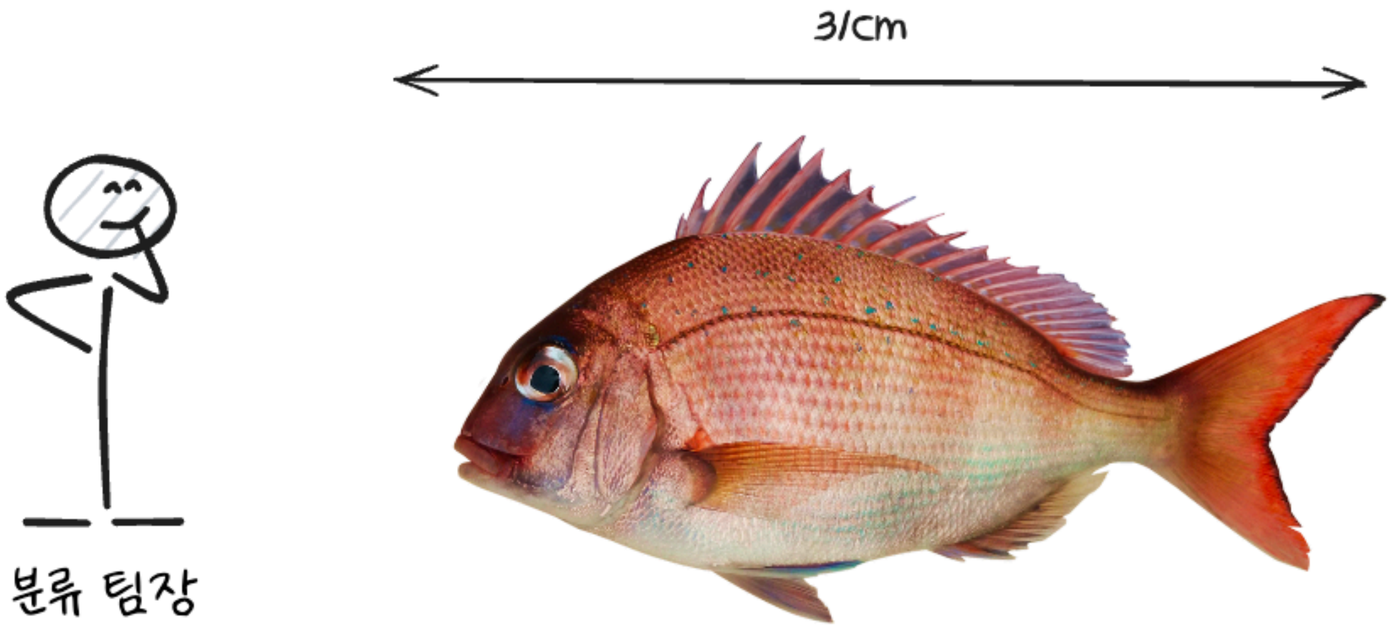
농어



빙어

사용된 데이터 셋은 캐글에 공개된 fish_market 데이터입니다. (https://www.kaggle.com/datasets/vipullrathod/fish-market)

단순하게 분류하기

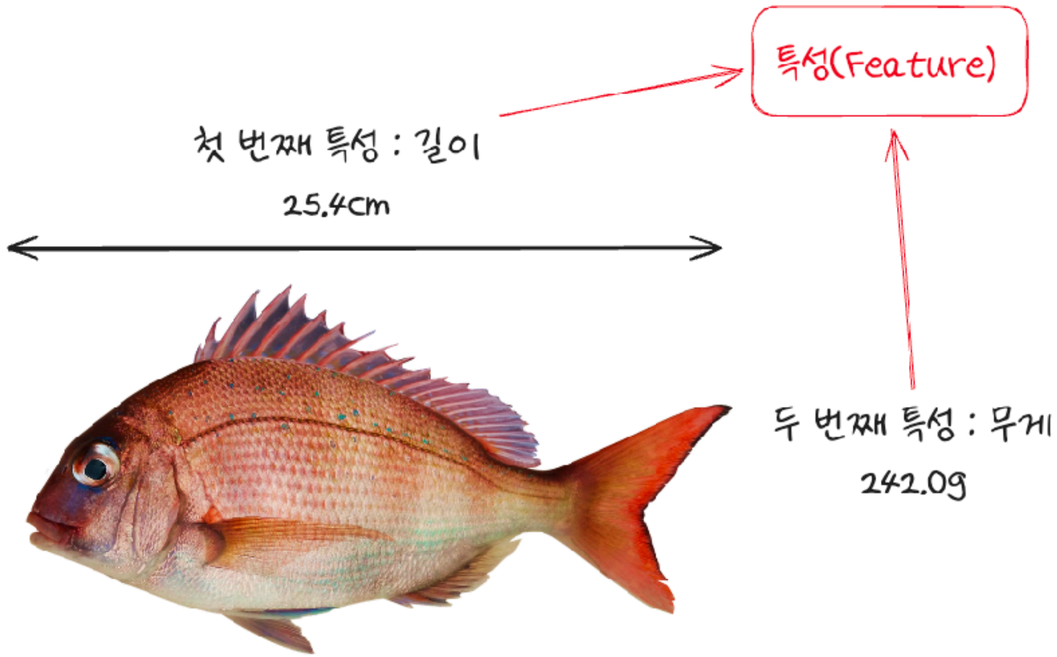



```
if fish_length >= 30:
  print("도미")
```

머신러닝 모델은?

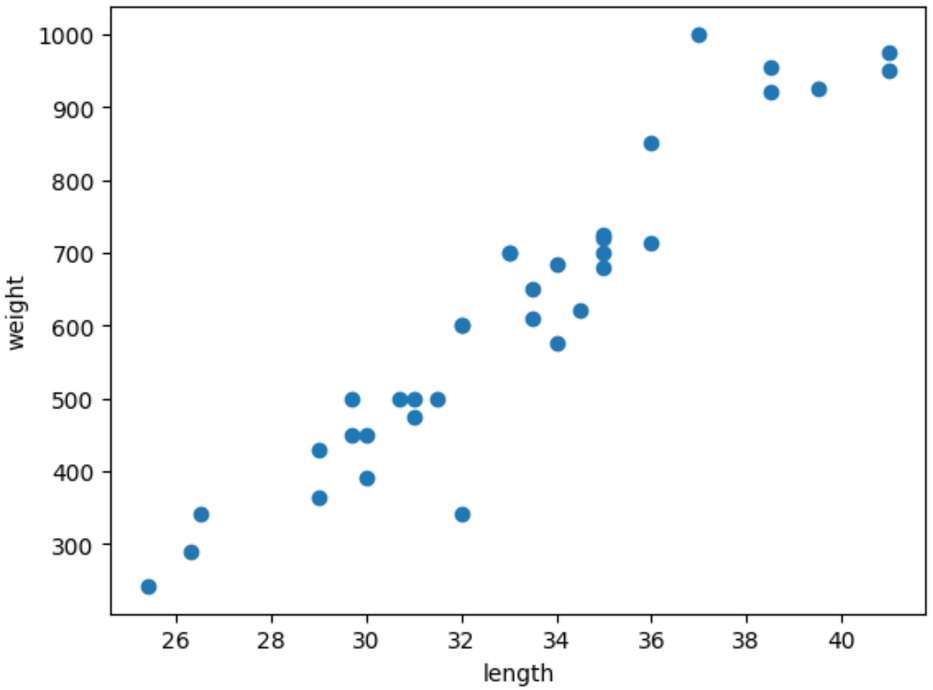
데이터 준비

```
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]
```



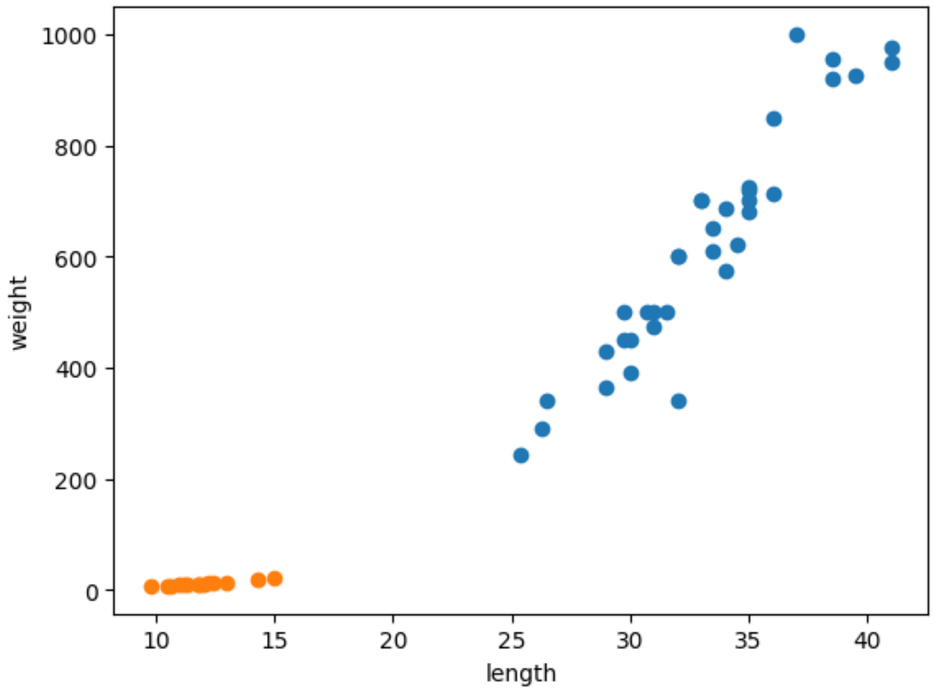
데이터 준비

```
import matplotlib.pyplot as plt

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



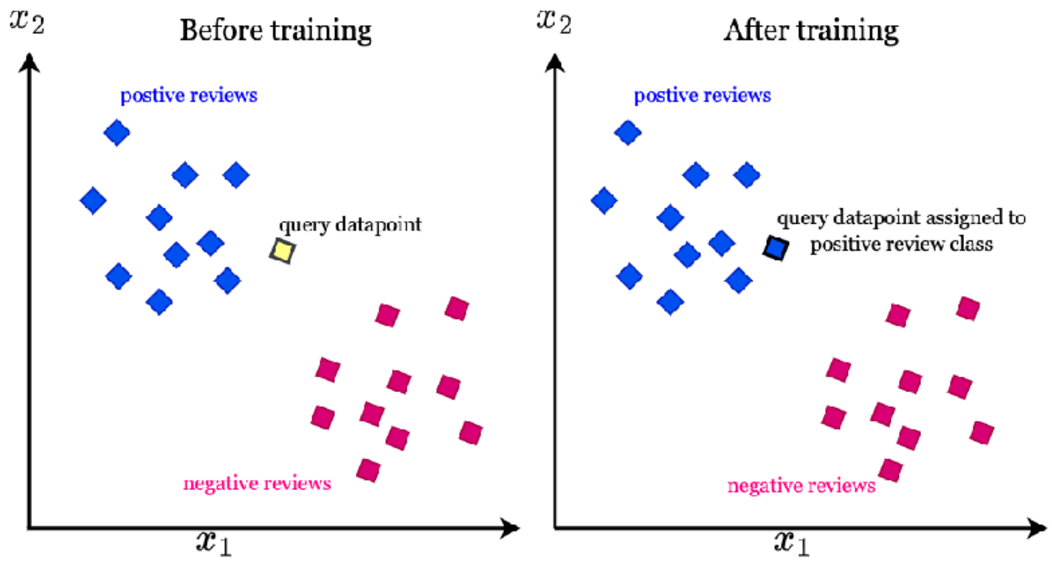
데이터 준비

```
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()
```



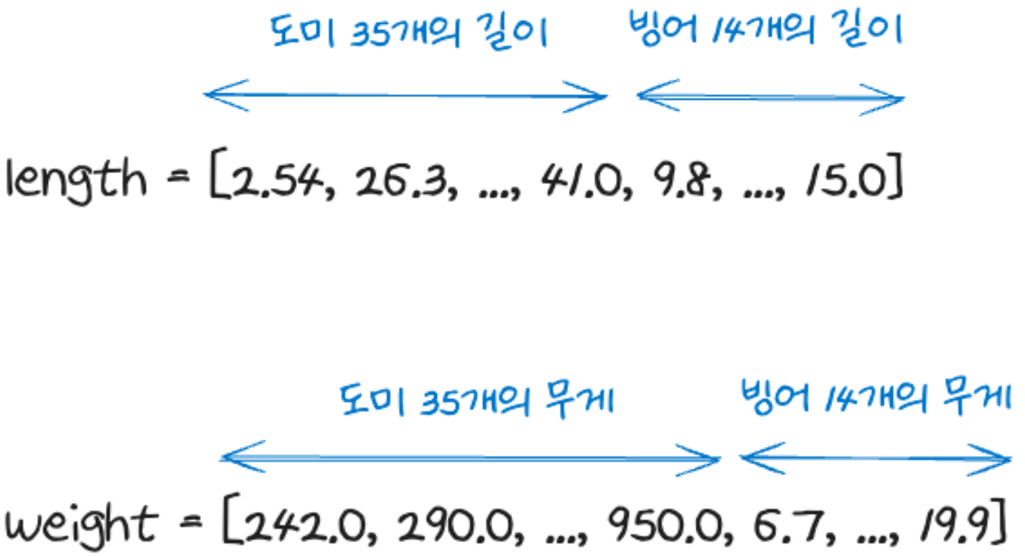
K 최근접 이웃(K nearest neighbors) 알고리즘

- 어떤 데이터에 대한 답을 구할 때 주위의 다른 데이터를 보고 다수를 차지하는 것을 정답으로 사용한다.



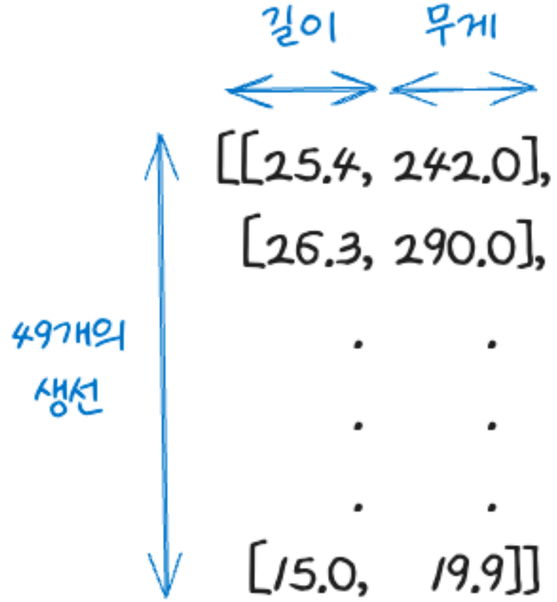
첫 머신러닝 프로그램

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



첫 머신러닝 프로그램

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



첫 머신러닝 프로그램

```
print(fish_data)

# [[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]]
```

첫 머신러닝 프로그램

```
fish_target = [1]*35 + [0]*14
print(fish_target)

# [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- 무게와 길이를 보고 도미와 빙어를 구분하는 규칙을 찾길 원한다.
- 어떤 생선이 도미인지 빙어인지를 알려줘야한다.
- 컴퓨터 프로그램은 문자를 직접 이해하지 못하므로 도미와 빙어를 숫자 1, 0으로 표현해야한다.

첫 머신러닝 프로그램

더 자세한 설명은 ChatGPT 설명을 참고하세요. (https://chat.openai.com/share/32b9287b-c50b-46c5-bbd6-c154892e69f0)

```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier()

# 학습
kn.fit(fish_data, fish_target)

# 평가
kn.score(fish_data, fish_target)
# 결과 : 1.0
```

첫 머신러닝 프로그램

더 자세한 설명은 ChatGPT 설명을 참고하세요. (https://chat.openai.com/share/32b9287b-c50b-46c5-bbd6-c154892e69f0)

```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier()

# 학습
kn.fit(fish_data, fish_target)

# 평가
kn.score(fish_data, fish_target)
# 결과 : 1.0
```

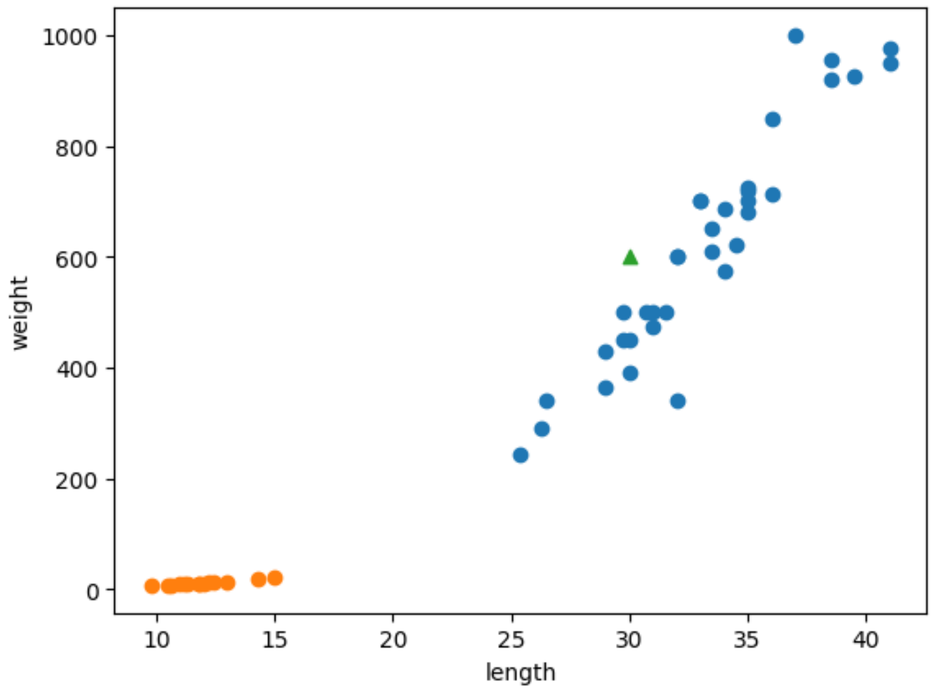
첫 머신러닝 프로그램

더 자세한 설명은 ChatGPT 설명을 참고하세요. (https://chat.openai.com/share/32b9287b-c50b-46c5-bbd6-c154892e69f0)

```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier()

# 학습
kn.fit(fish_data, fish_target)

# 평가
kn.score(fish_data, fish_target)
# 결과 : 1.0
```

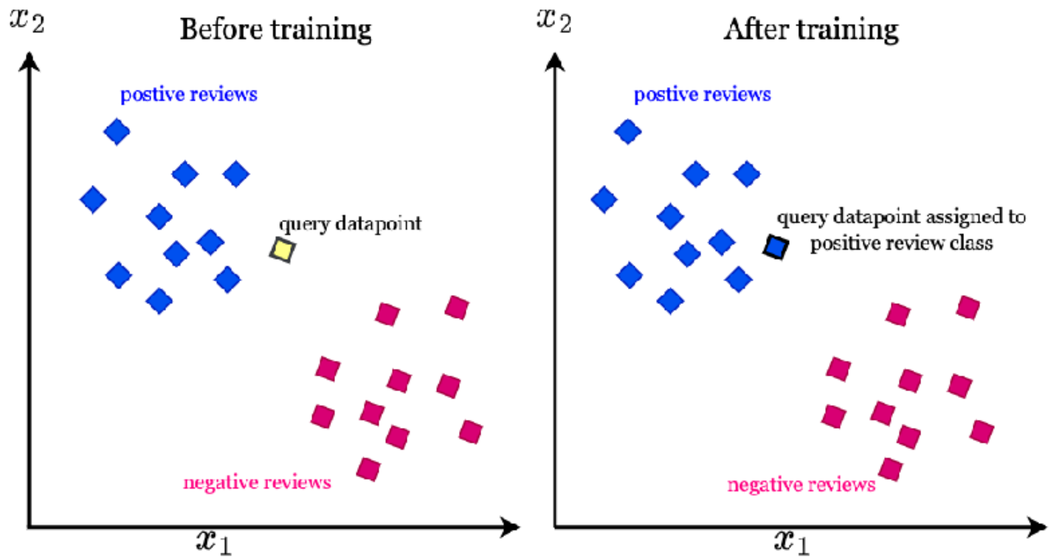
첫 머신러닝 프로그램

```
plt.scatter(bream_length,
bream_weight)
plt.scatter(smelt_length,
smelt_weight)
plt.scatter(30, 600, marker='^')
plt.xlabel('length')
plt.ylabel('weight')
plt.show()


kn.predict([[30, 600]])
# 결과 : array([1])

# kn =
KNeighborsClassifier(n_neighbors=49)로
확인할 이웃을 지정할 수 있다. 이 경우 도미가 49
마리중 35마리이므로 항상 도미로 판단한다.
```



K 최근접 이웃(K nearest neighbors) 알고리즘

- 이 알고리즘을 위해 준비할 일은 데이터를 메모리에 모두 올려놓는 것 뿐이다.
- 새로운 데이터 예측 시 가장 가까운 직선 거리에 어떤 데이터가 있는지 살펴보는 것 뿐이다.
- 그렇다보니 단점으로 데이터가 아주 많은 경우 메모리에 모든 점에 대한 계산을 다 올려놔야하기 때문에 메모리가 많이 필요하고, 시간이 오래 걸린다.



K 최근접 이웃(K nearest neighbors) 알고리즘

```
# 학습한 데이터를 메모리에 들고 있다.
print(kn._fit_X)

# 정답 데이터를 메모리에 들고 있다.
print(kn._y)
```

- 사이킷런의 KNeighborsClassifier 객체도 마찬가지로 학습한 데이터를 _fit_X에, 정답 데이터를 _y에 들고 있다. (확인해보자.)
- 즉 무언가 훈련되는게 없는 셈이다. fit() 메서드에 전달한 데이터를 모두 저장하고 있다가 새로운 데이터가 등장하면 가장 가까운 데이터를 참고하여 분류한다.

지도 학습

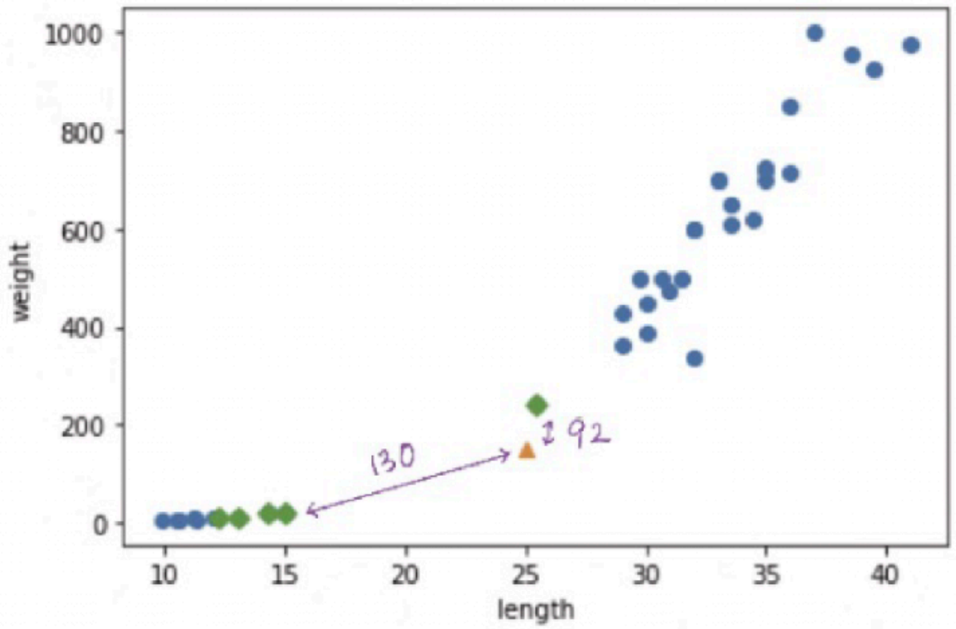
- 생선의 길이와 무게를 알고리즘에 사용했다. 이 경우 정답은 도미인지 아닌지 여부이다.
- 지도 학습에서는 데이터와 정답을 입력(input)과 타깃(target)이라하고 이 둘을 합쳐 훈련 데이터(training data)라고 부른다.
- 앞서 언급했듯 길이와 무게를 특성(feature)라고 부른다.
- 지도 학습은 정답(타깃)이 있으니 알고리즘이 정답을 맞히는 것을 학습한다.
- 앞서 만든 머신러닝 모델은 입력 데이터와 타깃을 사용했으므로 지도 학습 알고리즘이다.

Problem?

"이대로 끝내면 될까요?"

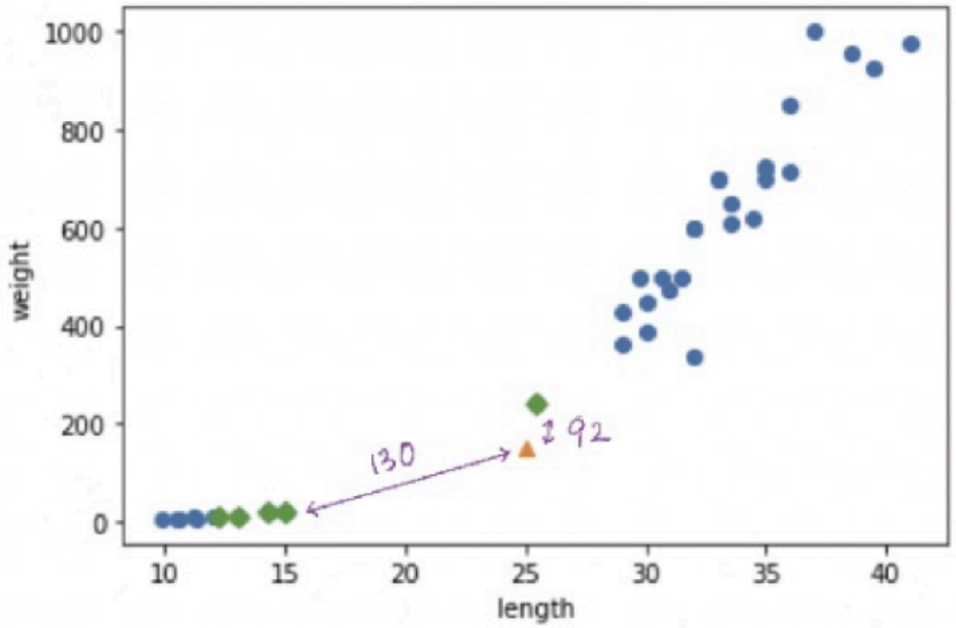
문제 제기

- 도미 35마리 빙어 14마리 모두 저장하고 못 맞추면 이상하잖아?
- 길이가 25cm이고 무게가 150g이면 도미인데 빙어로 예측하네?



문제 제기

- 도미 35마리 빙어 14마리 모두 저장하고 못 맞추면 이상하잖아? → 훈련 세트와 테스트 세트
- 길이가 25cm이고 무게가 150g이면 도미인데 빙어로 예측하네? → 데이터 전처리



E.O.D

개발 환경 설정

생선 분류 모델 만들기

훈련 세트 vs 테스트 세트

데이터 전처리

개발 환경 설정

생선 분류 모델 만들기

훈련 세트 vs 테스트 세트

데이터 전처리