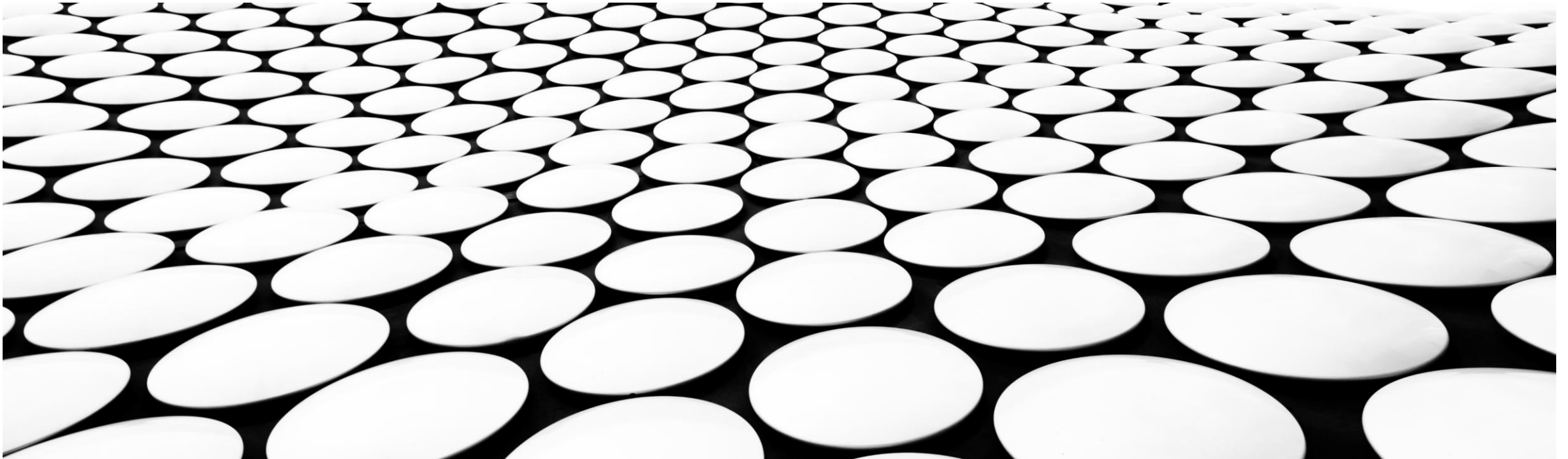


# 머신러닝을 쉽게 이해할 수 있을까요?

#분류문제 #머신러닝 #패키지 #훈련세트 #학습세트 #넘파이 #사이킷런



담당 강사 : 안상선 (sangsun.ahn@gmail.com)

# Chapter01 : 변수와 데이터



## 1.1 변수와 자료

- 변수(Variable) : 연구자의 관심대상이 되는 성격 or 속성
- 자료(Data) : 이러한 변수를 관찰하여 기록한 결과

한 집단의 특성을 쉽게 알아보고 분석하기 위해서는 수집된 자료를 의미 있는 모양으로 분류·정리하는 것이 중요하다.

## 1.2 자료의 종류

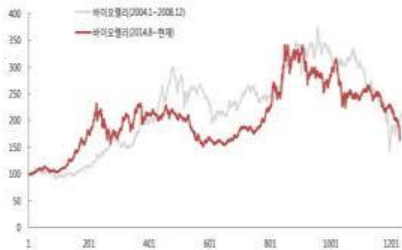
- 범주형 자료(질적, 비계량)
  - 명목변수(Nominal Variable) : 개체나 사람의 특성만 (성별, 종교 등)
  - 순서(서열)변수(Ordinal Variable) : 측정대상 간 선호를 부여 (선호도, 만족도)
- 양적인 자료(양적, 계량)
  - 등간변수(Interval Variable) : 측정대상 간의 순서 + 값 사이의 간격이 일정 (IQ, 온도)  
절대 영점이 없음(100도는 50도의 2배 뜨겁다?)
  - 비율변수(Ratio Variable) : 측정대상 간에 비율 계산이 가능함(연령, 무게, 거리, 시간)
  - 이산형 변수(Discrete Variable) : 점수, 빈도수
  - 연속형 변수(Continuous Variable) : 실수, 키, 몸무게

## 1.2 자료의 종류

### 연속형 변수와 이산형 변수

- 연속형 변수 : 시간에 대해서 행동과 상태가 유한차원 벡터공간  
: 주가 수익률, 자동차의 운행거리,
- 이산형 변수 : 시간에 대해서 행동과 상태가 원소형태의 값(Value)를 갖는다.  
: {라이트를 켜다, 라이트를 끄다, 매도한다, 매수한다}

연속형 변수 : 수익률



이산형 변수 : 전원 상태





## Chapter02 : 인공지능을 활용한 데이터 분석

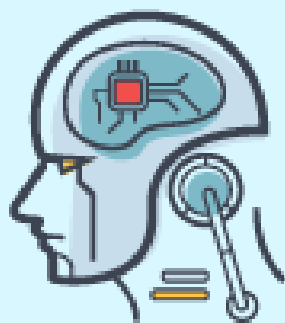


# 머신러닝과 딥러닝

Artificial Intelligence

## 인공지능

사고나 학습 등 인간이 가진  
지적 능력을 컴퓨터를 통해  
구현하는 기술



Machine Learning

## 머신러닝

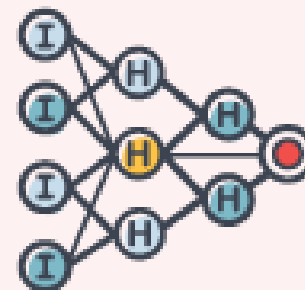
컴퓨터가 스스로 학습하여  
인공지능의 성능을  
향상 시키는 기술 방법



Deep Learning

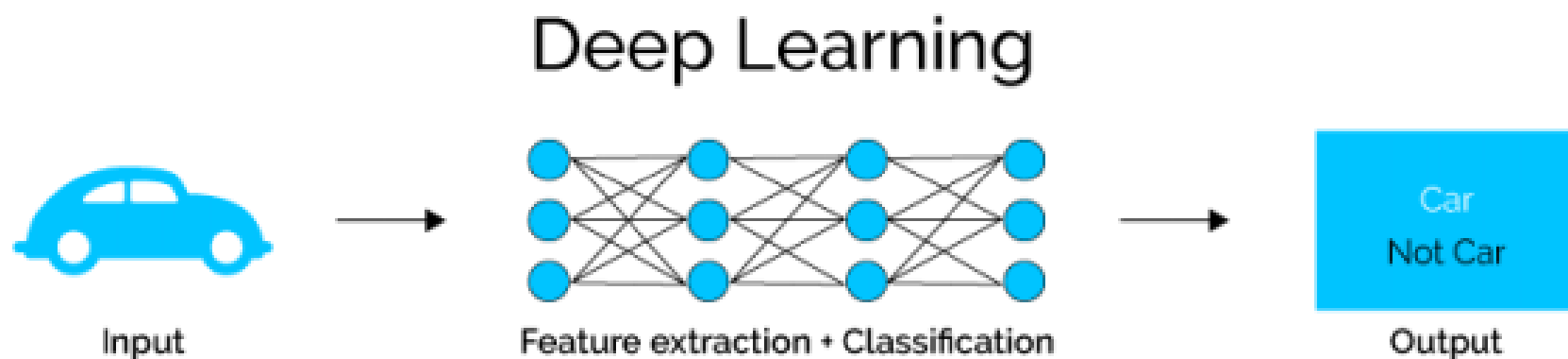
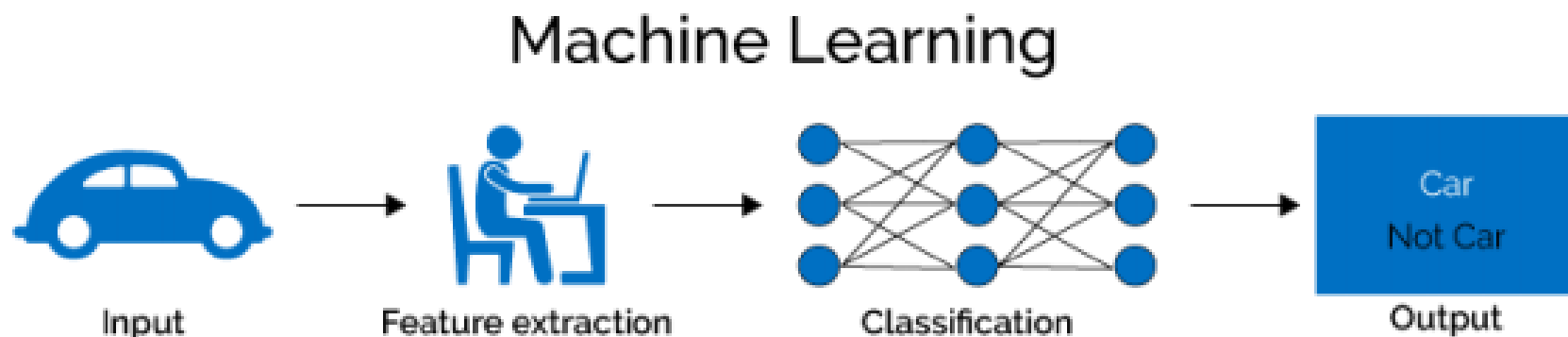
## 딥러닝

인간의 뉴런과 비슷한  
인공신경망 방식으로  
정보를 처리



# 머신러닝과 딥러닝

그림4 머신러닝 VS 딥러닝



자료: Towards Data Science, 메리츠증권증권 리서치센터



## 예시 : 신용카드 부도 예측

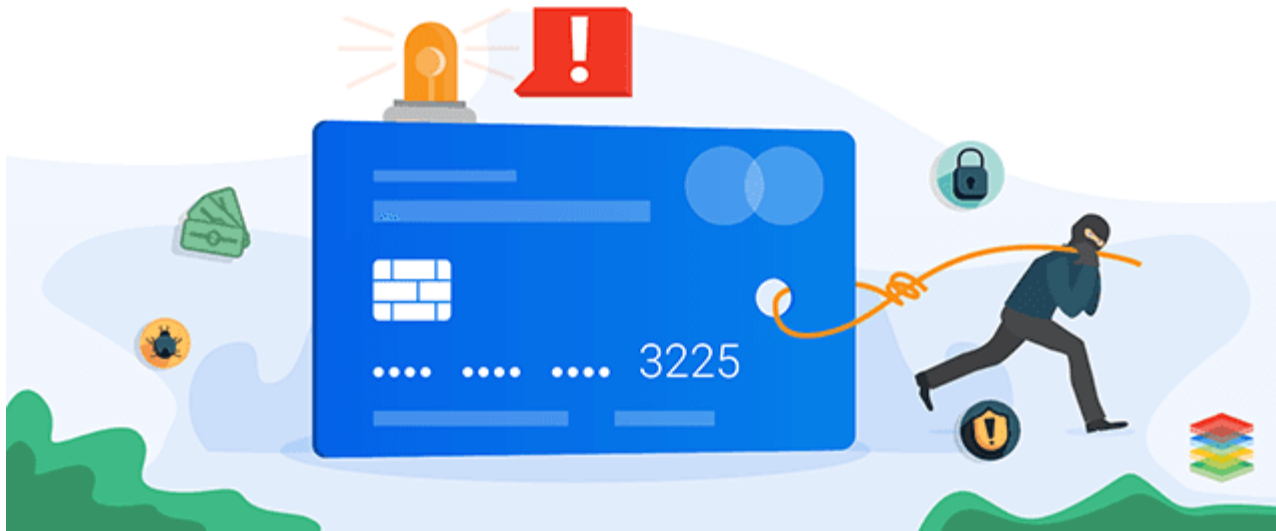


## 예시 : 강우량 예측



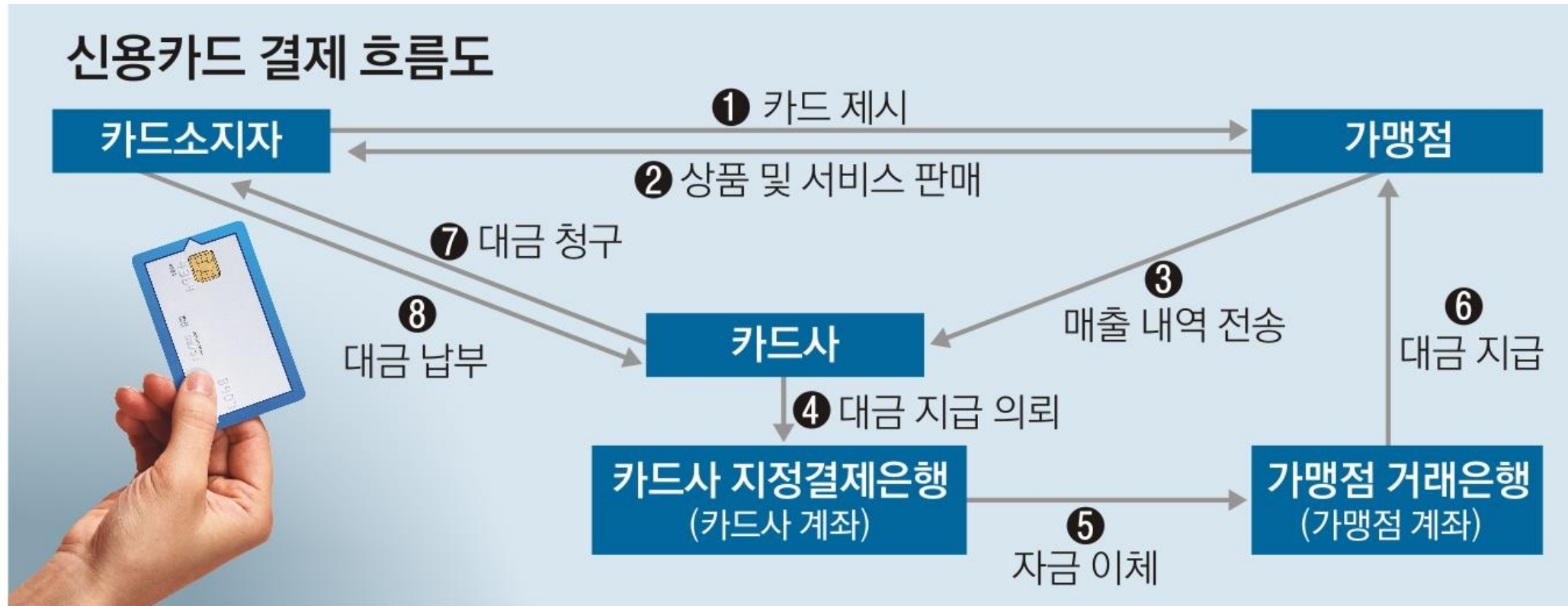
## 예시 : 신용카드 부도 예측

### Credit Card Fraud Detection



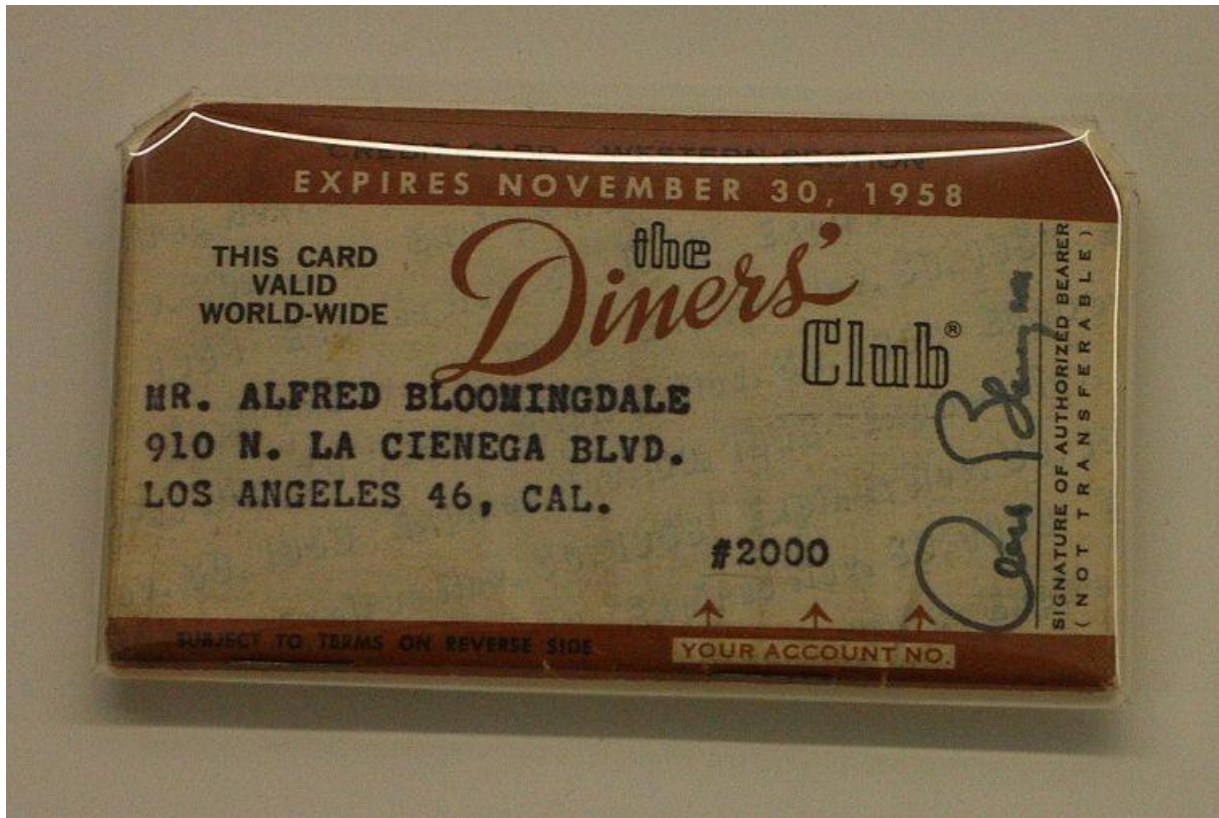
<https://www.kaggle.com/mlg-ulb/creditcardfraud>

# Chapter01 : 신용카드 부도예측





# Chapter01 : 신용카드 부도예측



# Chapter01 : 신용카드 부도예측



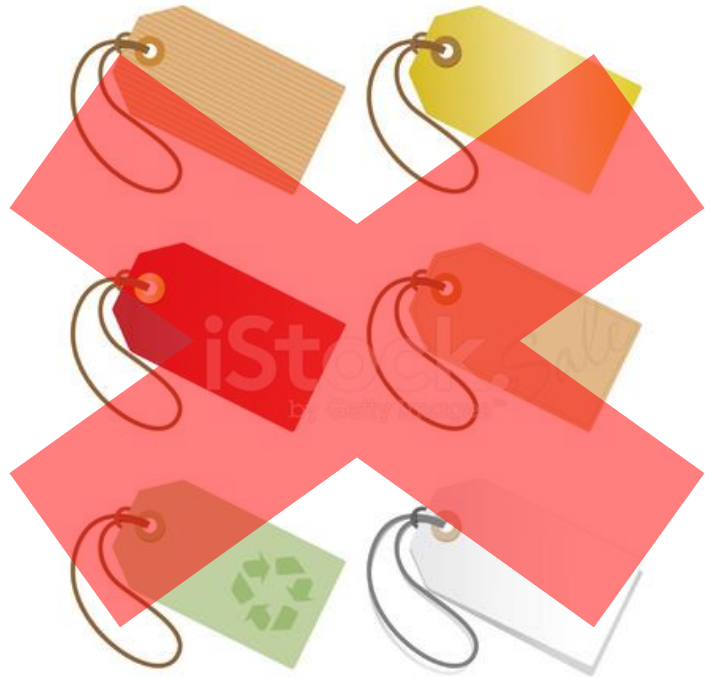


## Chapter02 : 생선분류 문제



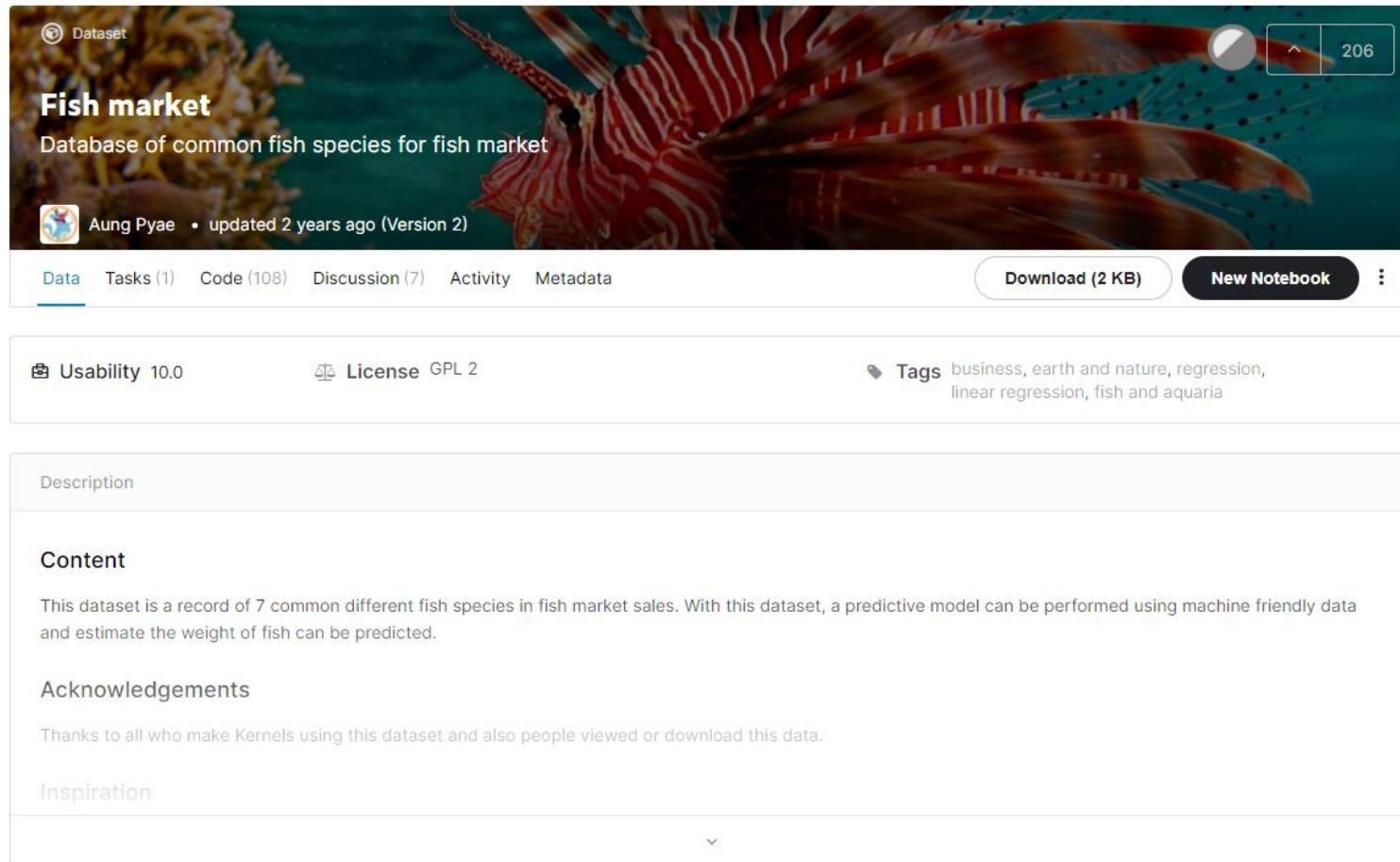
## 1) 생선 분류 문제

- 생선가게 : 생선은 **가격 태그**를 붙이기 어렵다.
- 편의점처럼 어떤 품목(생선종류)인지 또 가격(사이즈별 단가)을 알기 어려움



## 3> Data Set

- URL : <https://www.kaggle.com/aungpyaeap/fish-market>



The image shows the Kaggle dataset page for 'Fish market' by Aung Pyae. The header features a banner with a colorful fish and the text 'Fish market Database of common fish species for fish market'. Below the banner, there are tabs for 'Data', 'Tasks (1)', 'Code (108)', 'Discussion (7)', 'Activity', and 'Metadata'. To the right of the tabs are buttons for 'Download (2 KB)' and 'New Notebook'. Below the tabs, there is a section for 'Usability 10.0', 'License GPL 2', and 'Tags business, earth and nature, regression, linear regression, fish and aquaria'. The main content area has sections for 'Description', 'Content', 'Acknowledgements', and 'Inspiration'. The 'Content' section contains the text: 'This dataset is a record of 7 common different fish species in fish market sales. With this dataset, a predictive model can be performed using machine friendly data and estimate the weight of fish can be predicted.'

**Fish market**  
Database of common fish species for fish market

Aung Pyae • updated 2 years ago (Version 2)

[Data](#) [Tasks \(1\)](#) [Code \(108\)](#) [Discussion \(7\)](#) [Activity](#) [Metadata](#) [Download \(2 KB\)](#) [New Notebook](#)

**Usability** 10.0 **License** GPL 2 **Tags** business, earth and nature, regression, linear regression, fish and aquaria

**Description**

**Content**

This dataset is a record of 7 common different fish species in fish market sales. With this dataset, a predictive model can be performed using machine friendly data and estimate the weight of fish can be predicted.

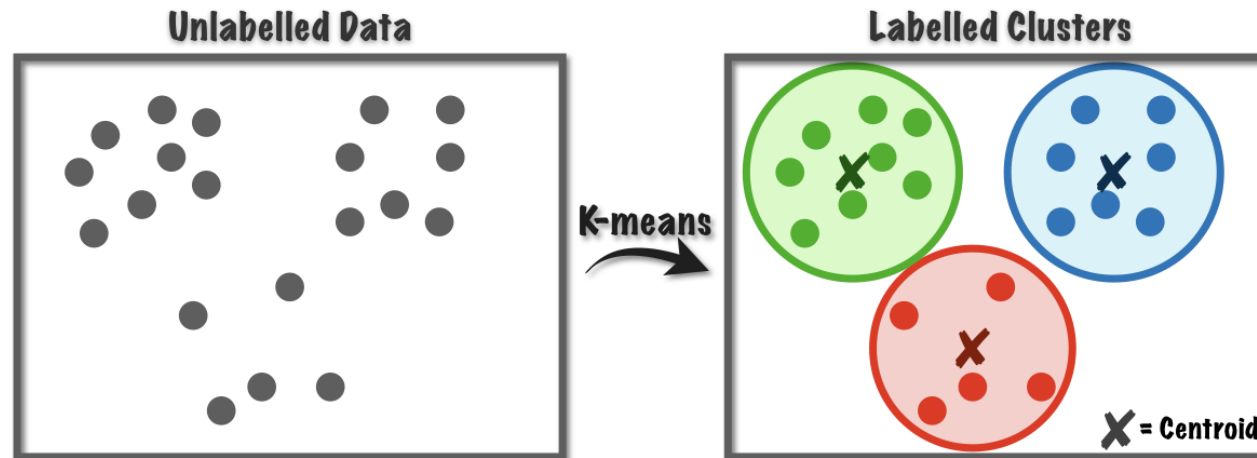
**Acknowledgements**

Thanks to all who make Kernels using this dataset and also people viewed or download this data.

**Inspiration**

## 4> 사용 알고리즘 : K-means clustering

- 주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로, 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 작동합니다.
- 이 알고리즘은 자율 학습의 일종으로, 레이블이 달려 있지 않은 입력 데이터에 레이블을 달아주는 역할을 수행합니다.





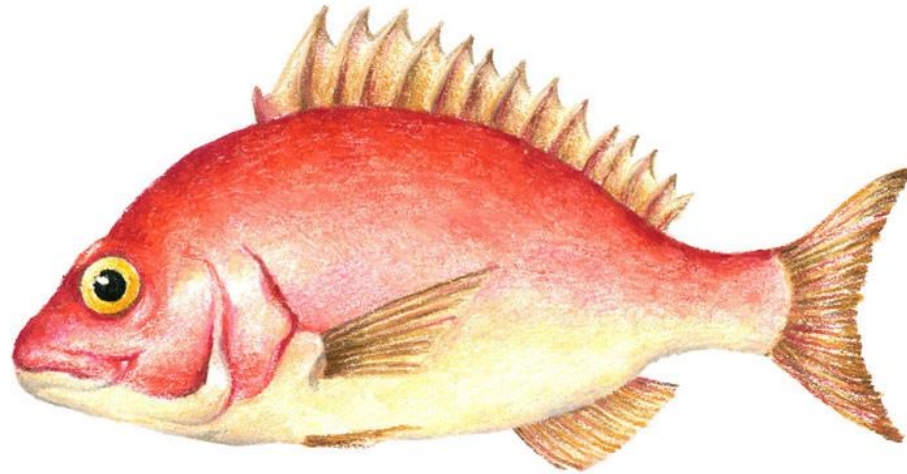
## 4> 사용 알고리즘 : ~~K-means clustering~~

하다 보면 늘어요!  
이제 눈 대중으로만 봐도  
품종, 사이즈, 가격을 맞춰요!



## 5> 1단계 : 조건 지정

생선 길이가 30cm 이상이면 면 **도미**입니다





## 5> 1단계 : 조건 지정

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



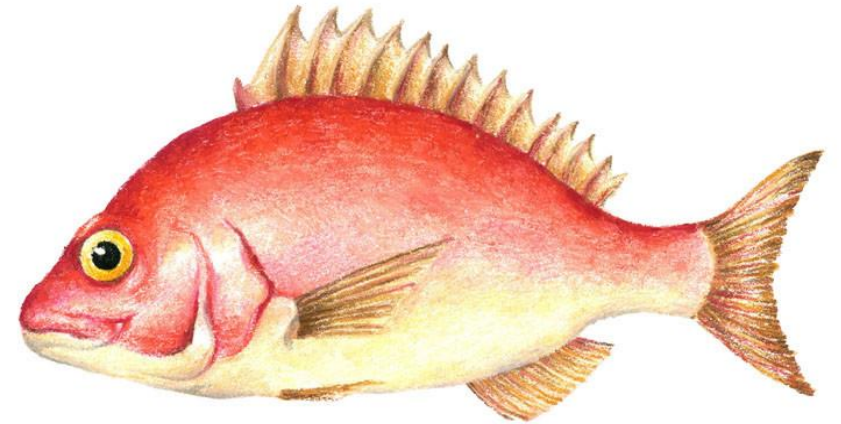
## 5> 1단계 : 조건 지정

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

생각해볼 문제 : **조건식만으로는 아무 것도 못한다**

무엇이 필요할까요?

- 1> 생선 데이터가 필요합니다.
- 2> 생선 데이터를 불러와야 합니다.
- 3> 생선 데이터 중에 길이 값이 필요합니다.
- 4> 생선 데이터 중에 길이 값 단위는 "cm"
- 5> 생선 데이터 중에 길이 값의 이름은 fish\_length



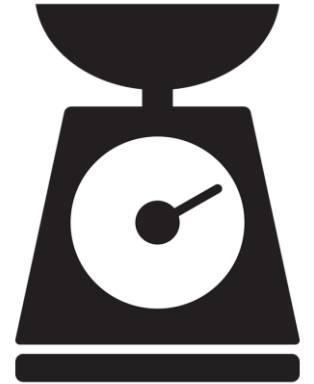
## 5> 1단계 : 조건 지정

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

**또** 생각해볼 문제 :  
**하나의 조건**만으로는 판단하기 어렵다.

생선의 길이 외에 다른 조건이 필요하다.

+ 생선의 무게(kg)도 고려함




## 5> 1단계 : 조건 지정



## 6> 데이터 입력

### - 코랩에서 방 이름 바꾸기



Colaboratory에 오신 것을 환영합니다

파일 수정 보기 삽입 런타임 도구 도움말

새 노트  
노트 열기 Ctrl+O  
노트 업로드  
이름 바꾸기  
드라이브에 사본 저장  
GitHub Gist로 사본 저장  
GitHub에 사본 저장  
저장 Ctrl+S  
업데이트 기록  
.ipynb 다운로드  
.py 다운로드  
드라이브 미리보기 업데이트  
인쇄 Ctrl+P

코드 + 텍스트 Drive로 복사

### Colaboratory란?

줄여서 'Colab'이라고도 하는 Colaboratory를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있습니다. Colab은 다음과 같습니다.

- 구성이 필요하지 않음
- GPU 무료 액세스
- 간편한 공유

학생이든, 데이터 과학자든, AI 연구원이든 Colab으로 업무를 더욱 간편하게 처리할 수 있습니다. [Colab 소개 영상](#)에서 자세한 내용을 아래에서 시작해 보세요.

### 시작하기

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

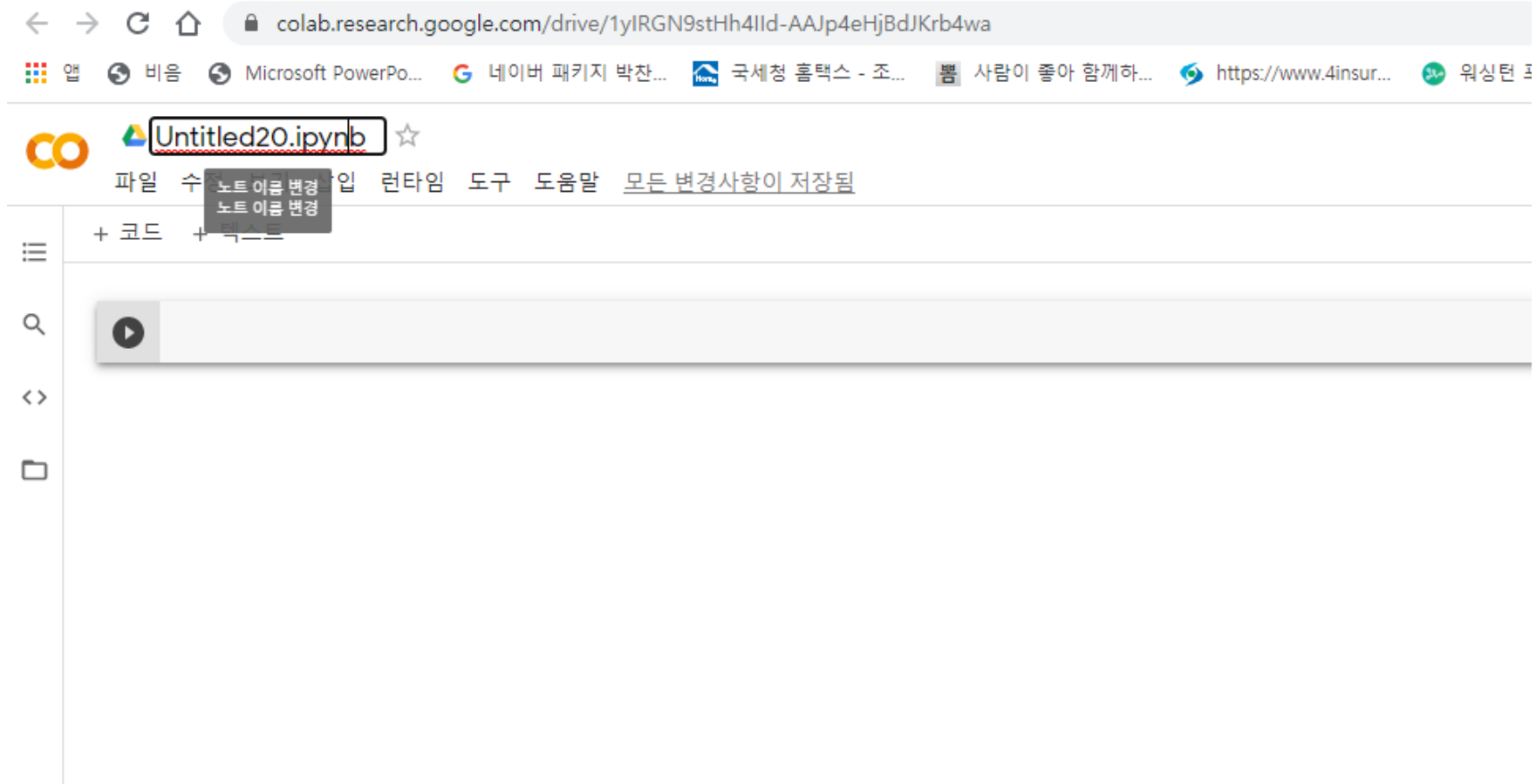
예를 들어 다음은 값을 계산하여 변수로 저장하고 결과를 출력하는 간단한 Python 스크립트가 포함된 코드 셀입니다.

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

-----
```

## 6> 데이터 입력

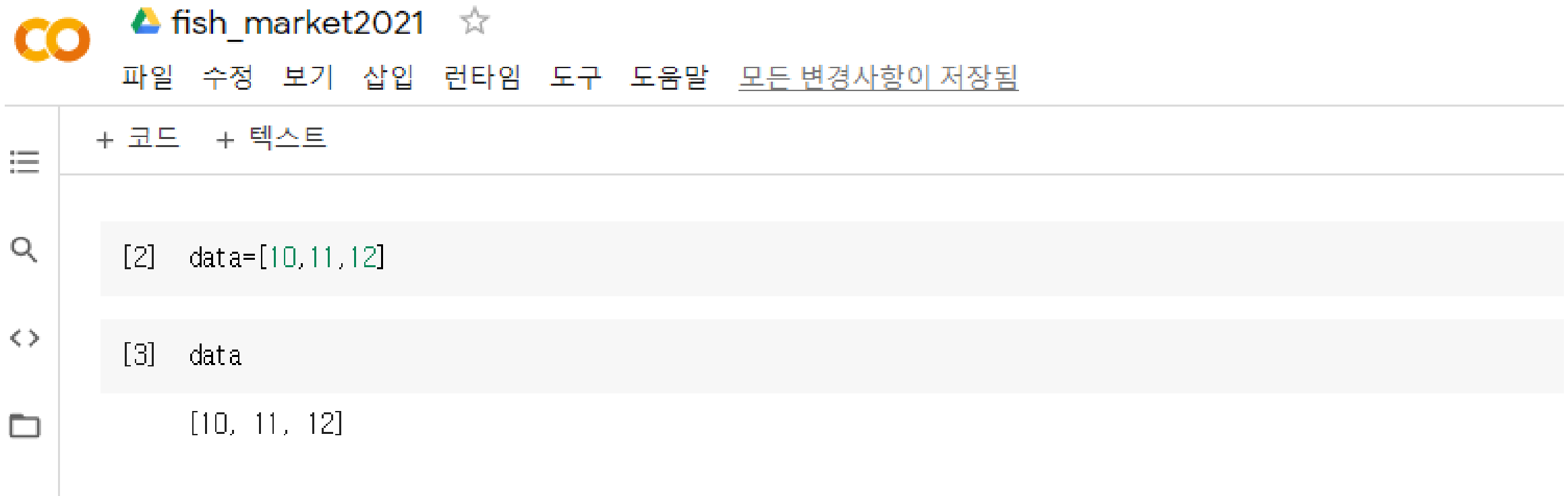
### - 코랩에서 방만들기





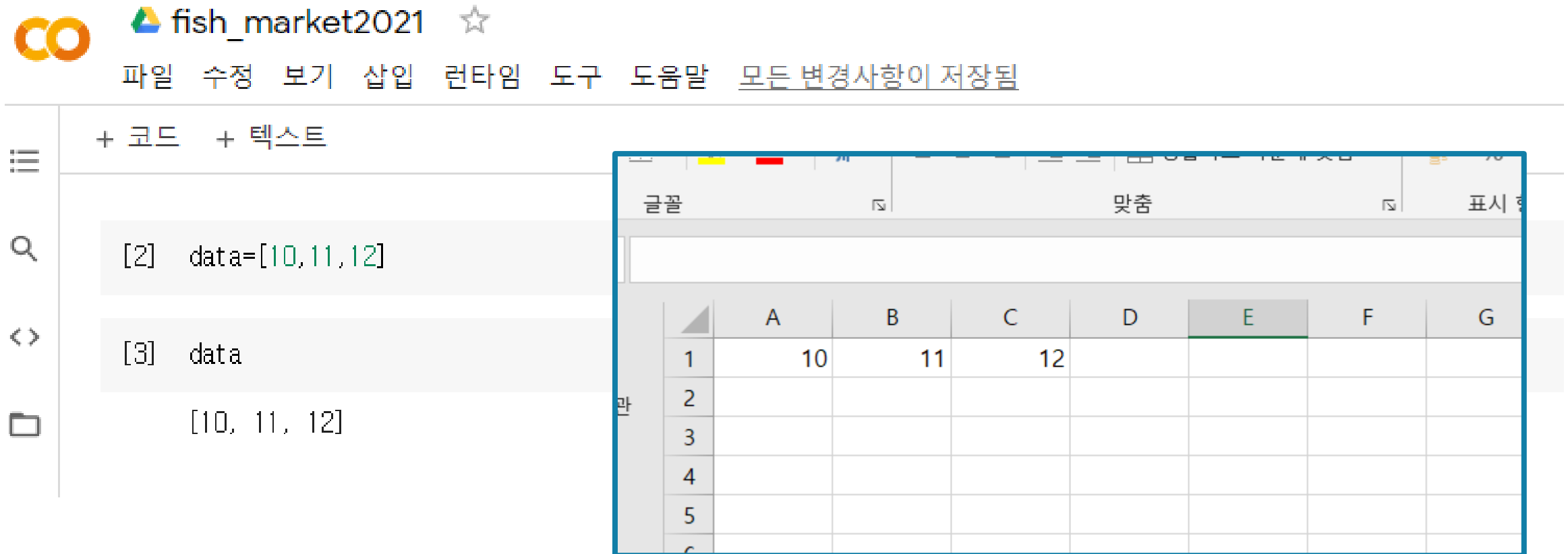
## 6> 데이터 입력

### - 파이썬에서 데이터 입력



## 6> 데이터 입력

### - 파이썬에서 데이터 입력



The screenshot shows a Google Colab interface. At the top, the title bar reads "fish\_market2021" with a star icon. Below it, a menu bar contains "파일", "수정", "보기", "삽입", "런타임", "도구", "도움말", and "모든 변경사항이 저장됨". On the left, a sidebar shows a file explorer with a folder icon and a code editor with a search icon. The code editor contains the following Python code:

```
[2] data=[10,11,12]
```

```
[3] data
```

```
[10, 11, 12]
```

Overlaid on the right side of the code editor is a data table. The table has a header row with columns labeled A, B, C, D, E, F, and G. The first row of data contains the values 10, 11, and 12 in columns A, B, and C respectively. The rest of the table is empty.

|   | A  | B  | C  | D | E | F | G |
|---|----|----|----|---|---|---|---|
| 1 | 10 | 11 | 12 |   |   |   |   |
| 2 |    |    |    |   |   |   |   |
| 3 |    |    |    |   |   |   |   |
| 4 |    |    |    |   |   |   |   |
| 5 |    |    |    |   |   |   |   |
| 6 |    |    |    |   |   |   |   |

## 6> 데이터 입력

### - 파이썬에서 데이터 입력

: 아래 값을 `bream_length`라는 변수로 입력해 보세요

[23.2, 24, 23.9, 26.3, 26.5, 26.8, 26.8, 27.6, 27.6, 28.5, 28.4, 28.7, 29.1,  
29.5, 29.4, 29.4, 30.4, 30.4, 30.9, 31, 31.3, 31.4, 31.5, 31.8, 31.9, 31.8, 32,  
32.7, 32.8, 33.5, 35, 35, 36.2, 37.4, 38]

\* `bream(도미)`

## 6> 데이터 입력

### - 파이썬에서 데이터 입력

: 아래 값을 `bream_weight`라는 변수로 입력해 보세요

[242, 290, 340, 363, 430, 450, 500, 390, 450, 500, 475, 500, 500,  
340, 600, 600, 700, 700, 610, 650, 575, 685, 620, 680, 700, 725,  
720, 714, 850, 1000, 920, 955, 925, 975, 950]

\* `bream(도미)`

## 6> 데이터 입력

- 무게와 크기를 입력한 값이 모두 몇 개인가요?

명령어 : `len(데이터 셀 이름)`

입력 값

```
len(bream_length)  
len(bream_weight)
```

## 6> 데이터 입력 (아웃풋)

```
[9] bream_length=[23.2, 24, 23.9, 26.3, 26.5, 26.8, 26.8, 27.6, 27.6, 28.5, 28.4, 28.7, 29.1, 29.5, 29.4, 29.4, 30.4, 30.4, 30.9, 31, 31.3, 31.4, 31.5, 31.8, 31.9, 31.8, 32, 32.7, 32.8, 33.5, 35, 35, 36.2, 37.4, 38]
```

▶ bream\_length

↳ [23.2,  
24,  
23.9,  
26.3,  
26.5,  
26.8,  
26.8,  
27.6,  
27.6,  
28.5,  
28.4,  
28.7,  
29.1,  
29.5,  
29.4,  
29.4,  
30.4,  
30.4,  
30.9,  
31,  
31.3,  
31.4,  
31.5,  
31.8,  
31.9,  
31.8,  
32,  
32.7,  
32.8,  
33.5,  
35,  
35,  
36.2,  
37.4,  
38]



## 6> 데이터 입력 (아웃풋)

[14] bream\_weight

```
[242,  
290,  
340,  
363,  
430,  
450,  
500,  
390,  
450,  
500,  
475,  
500,  
500,  
340,  
600,  
600,  
700,  
700,  
610,  
650,  
575,  
685,  
620,  
680,  
700,  
725,  
720,  
714,  
850,  
1000,  
920,  
955,  
925,  
975,  
950]
```

## 6> 데이터 입력 (아웃풋)

```
[15] len(bream_length)
```

```
35
```

```
[16] len(bream_weight)
```

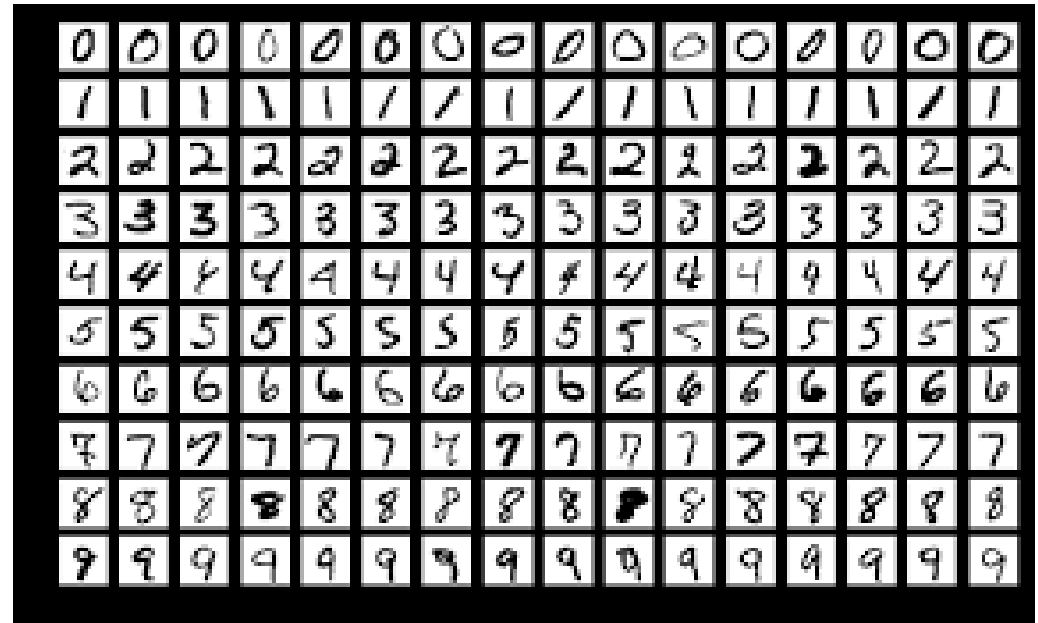
```
35
```

## 7> 데이터를 표현해 보기 (아웃풋)

데이터를 보겠습니다. 잘 보이십니까?

|    | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|----|-------------------|------------------|-------------------|------------------|--------|
| 0  | 5.1               | 3.5              | 1.4               | 0.2              | 0.0    |
| 1  | 4.9               | 3.0              | 1.4               | 0.2              | 0.0    |
| 2  | 4.7               | 3.2              | 1.3               | 0.2              | 0.0    |
| 3  | 4.6               | 3.1              | 1.5               | 0.2              | 0.0    |
| 4  | 5.0               | 3.6              | 1.4               | 0.2              | 0.0    |
| 5  | 5.4               | 3.9              | 1.7               | 0.4              | 0.0    |
| 6  | 4.6               | 3.4              | 1.4               | 0.3              | 0.0    |
| 7  | 5.0               |                  |                   | 0.2              | 0.0    |
| 8  | 4.4               | 2.9              | 1.4               | 0.2              | 0.0    |
| 9  | 4.9               |                  |                   | 0.1              | 0.0    |
| 10 | 5.4               | 3.7              | 1.5               | 0.2              | 0.0    |
| 11 | 4.8               | 3.4              | 1.6               | 0.2              | 0.0    |
| 12 | 4.8               | 3.0              | 1.4               | 0.1              | 0.0    |
| 13 | 4.3               | 3.0              | 1.1               | 0.1              | 0.0    |
| 14 | 5.6               | 4.0              | 1.2               | 0.2              | 0.0    |

**4 Features**  
**4 Dimensions**



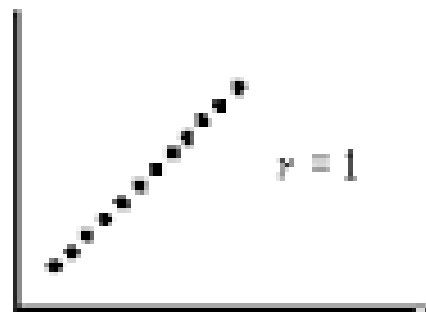
## 7> 데이터를 표현해 보기 (아웃풋)

데이터를 '한 눈'에 보고 싶습니다 : 시각화

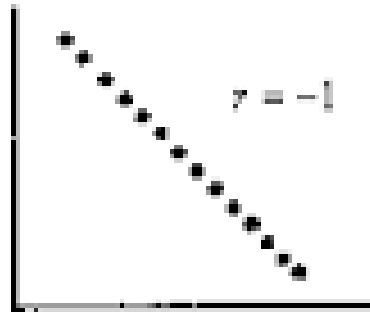


## 7> 데이터를 표현해 보기 (아웃풋)

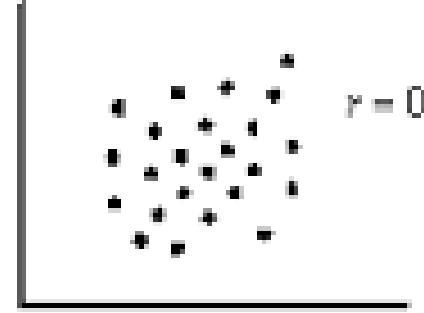
도미의 무게(x)와 크기(y)의 관계를 보겠습니다!



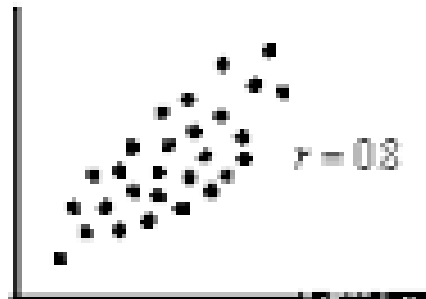
완전 正(+)상관



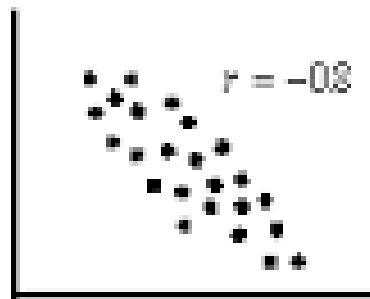
완전 負(-)상관



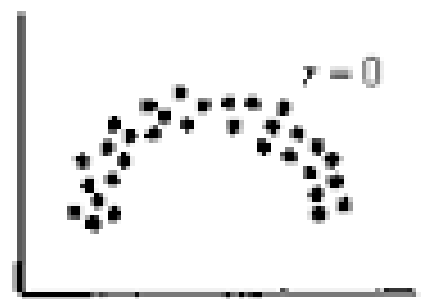
완전 無상관



正(+)상관



負(-)상관



無상관

## 7> 데이터를 표현해 보기 (아웃풋)

```
plt.scatter(bream_length, bream_weight)
```

```
plt.xlabel('length')
```

```
plt.ylabel('weight')
```

```
plt.show()
```



## 7> 데이터를 표현해 보기 (아웃풋)

```
import matplotlib.pyplot as plt
```

```
#함수를 부릅니다.
```

```
# 그리고 matplotlib의 pyplot함수를 plt로 줄여서 사용
```

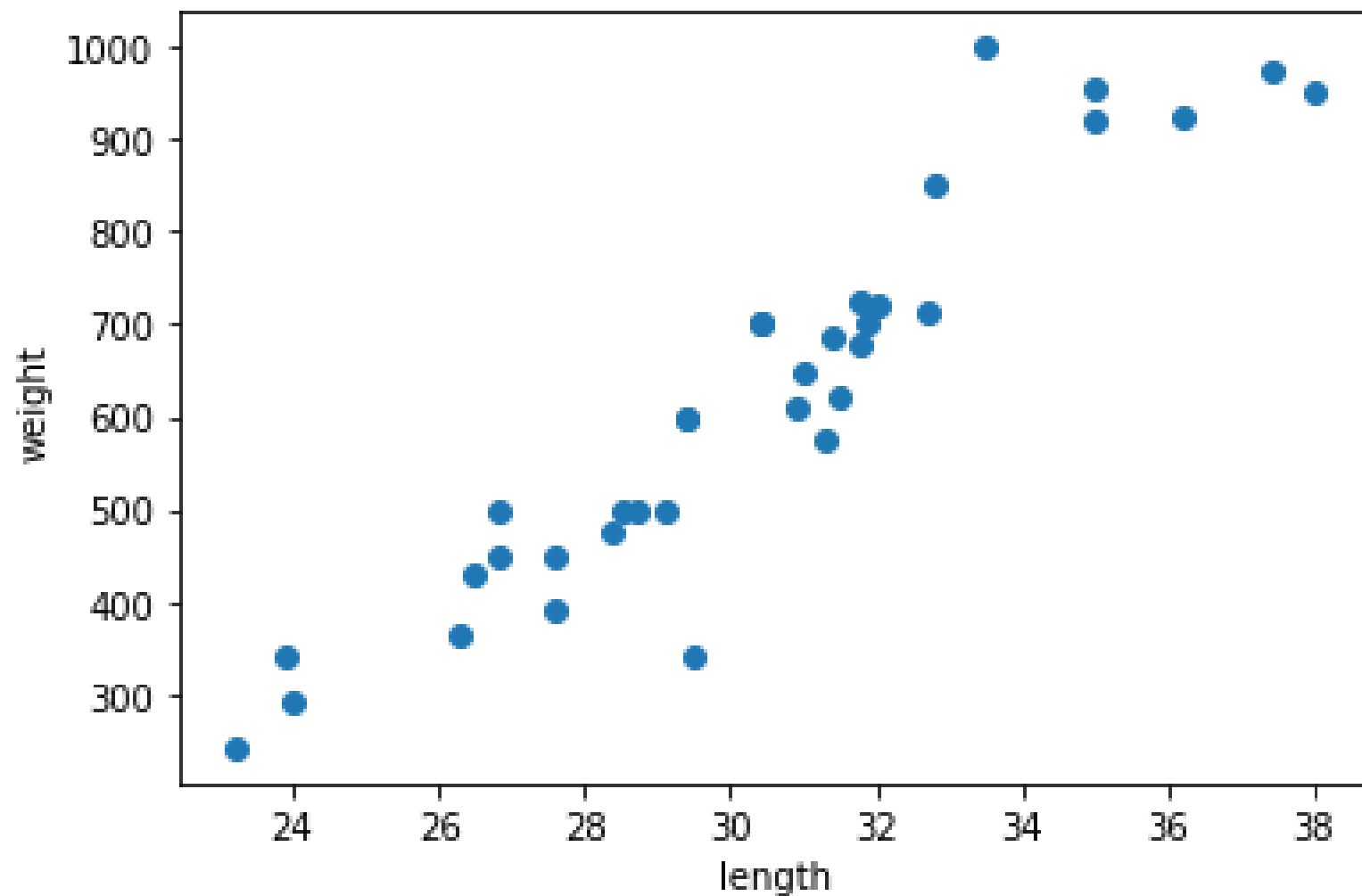
```
plt.scatter(bream_length, bream_weight)
```

```
plt.xlabel('length')
```

```
plt.ylabel('weight')
```

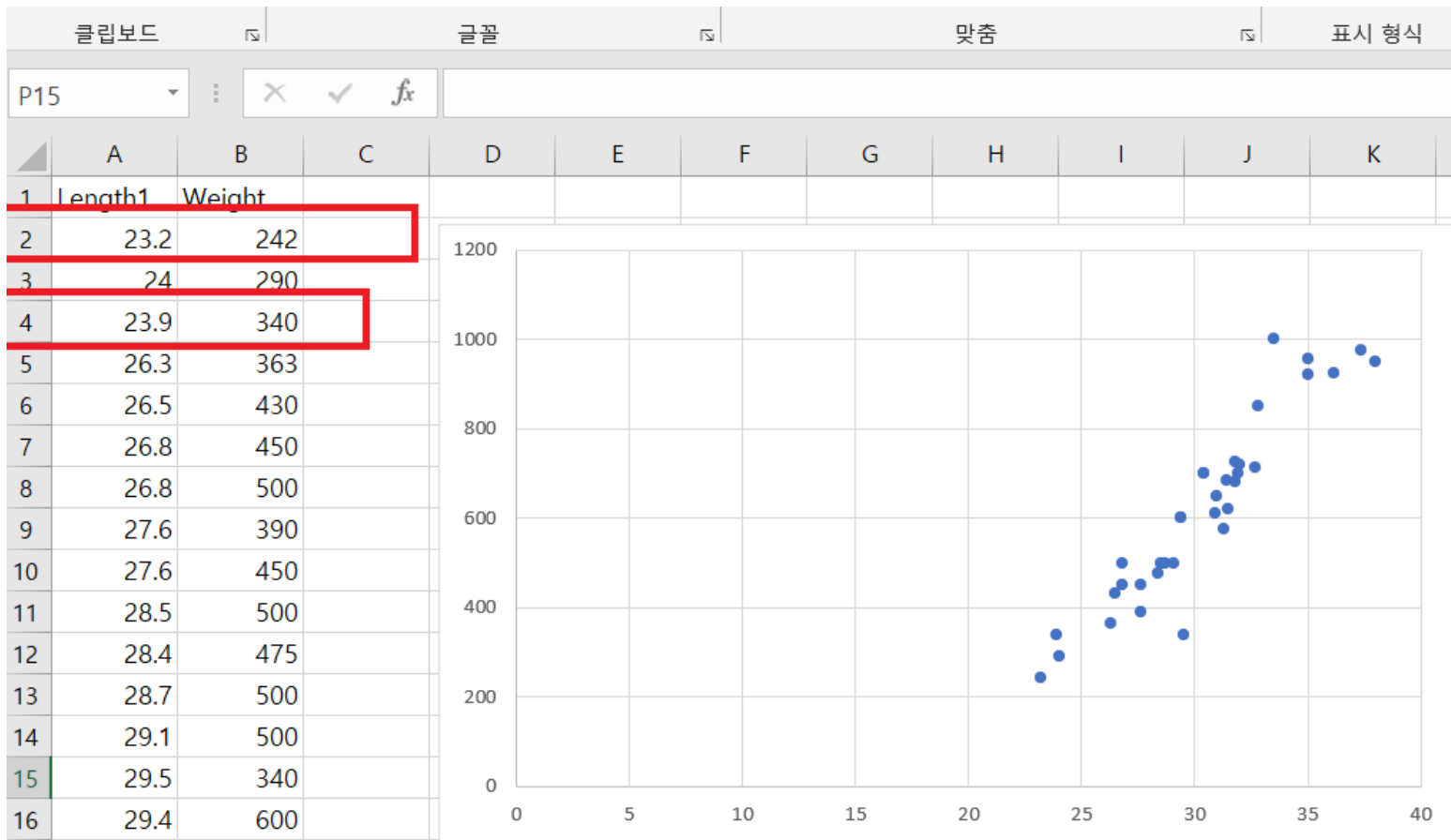
```
plt.show()
```

## 7> 데이터를 표현해보기



## 7> 데이터를 표현해 보기

- 실제 작업 : 데이터를 pair로 묶고 좌표를 찍음



## 7> 추가 데이터 입력 및 시각화

- 빙어(smelt) 데이터 입력 : 총 14마리

- 빙어의 길이(length)

9.3, 10, 10.1, 10.4, 10.7, 10.8, 11.3, 11.3, 11.4, 11.5, 11.7,

12.1, 13.2, 13.8

- 빙어의 무게(weight)

6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10, 9.9, 9.8, 12.2, 13.4, 12.2,

19.7, 19.9

## 8> 데이터 표현해 보기 (아웃풋)

```
[25] smelt_length=[9.3, 10, 10.1, 10.4, 10.7, 10.8, 11.3, 11.3, 11.4, 11.5, 11.7, 12.1, 13.2, 13.8]  
len(smelt_length)
```

14

```
[29] 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]  
len(smelt_weight)
```

14

## 8> 데이터 표현해 보기 (아웃풋)

```
[25] smelt_length=[9.3, 10, 10.1, 10.4, 10.7, 10.8, 11.3, 11.3, 11.4, 11.5, 11.7, 12.1, 13.2, 13.8]  
len(smelt_length)
```

14

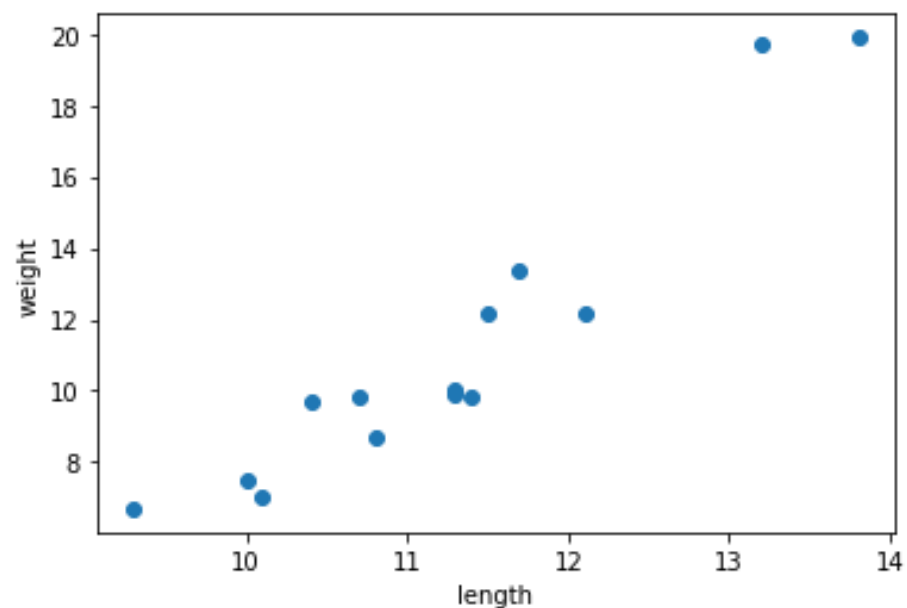
```
[29] smelt_weight=[6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4, 12.2, 19.7, 19.9]  
len(smelt_weight)
```

14

## 8> 데이터 표현해 보기 (아웃풋)

```
[31] import matplotlib.pyplot as plt #matplotlib의 pyplot함수를 plt로 줄여서 사용
```

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





## 9> 머신러닝 프로그램을 위한 데이터 합치기

### #1 데이터 합치기

`length=breame_length+smelt_length`

`weight=breame_weight+smelt_weight`

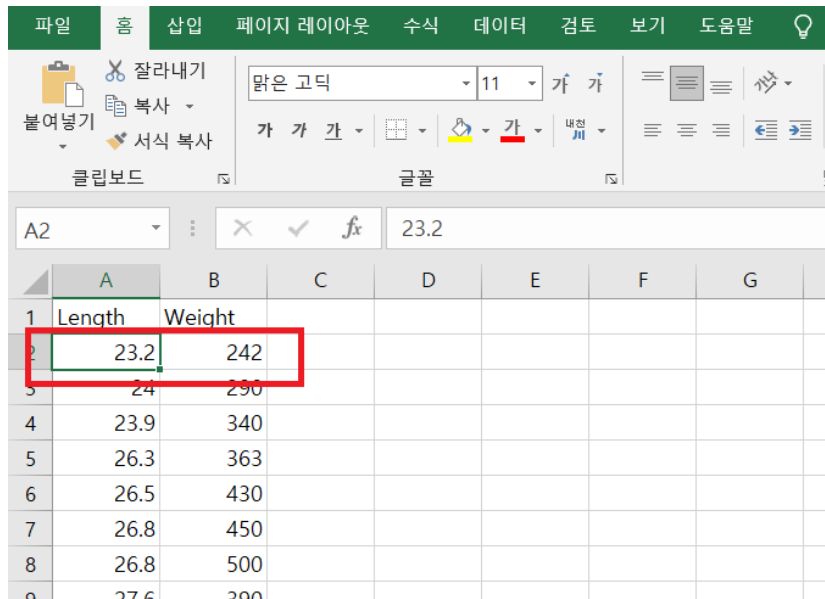
### #2 합친 개수 확인

`len(length), len(weight)`

## 9> 머신러닝 프로그램을 위한 데이터 합치기

### #3 데이터 합치기

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



The screenshot shows the Microsoft Excel interface. The active cell is A2, containing the value 23.2. The formula bar shows the value 23.2. The data table is as follows:

|   | A      | B      | C | D | E | F | G |
|---|--------|--------|---|---|---|---|---|
| 1 | Length | Weight |   |   |   |   |   |
| 2 | 23.2   | 242    |   |   |   |   |   |
| 3 | 24     | 290    |   |   |   |   |   |
| 4 | 23.9   | 340    |   |   |   |   |   |
| 5 | 26.3   | 363    |   |   |   |   |   |
| 6 | 26.5   | 430    |   |   |   |   |   |
| 7 | 26.8   | 450    |   |   |   |   |   |
| 8 | 26.8   | 500    |   |   |   |   |   |
| 9 | 27.6   | 500    |   |   |   |   |   |

## 9> 머신러닝 프로그램을 위한 데이터 합치기

### #2 결과

```
[38] length=bream_length+smelt_length  
weight=bream_weight+smelt_weight  
len(length), len(weight)
```

(49, 49)

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

```
[42] print(fish_data)
```

```
[[23.2, 242], [24, 290], [23.9, 340], [26.3, 363], [26.5, 430], [26.8, 450], [26.8, 500], [27.6, 390], [27.6, 450], [28.5, 500], [28.4, 475], [28.7, 500], [29.
```

## 9> 머신러닝 프로그램을 위한 데이터 합치기

### # 4 라벨링

1번 부터 35번은 1번(도미), 36번 부터 49번까지는 0번(빙어)

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

```
print(fish_target)
```

## #4 라벨링

[ [23.2, 242], [24, 290], [23.9, 340], [26.3, 363], [26.5, 430], [26.8, 450], [26.8, 500], [27.6, 390], [27.6, 450], [28.5, 500], [28.4, 475], [28.7, 500], [29.1, 5

[illegible]

## 10> 머신러닝

### \*컨셉

|    | A      | B      | C      | D | E | F |
|----|--------|--------|--------|---|---|---|
| 1  | Length | Weight | target |   |   |   |
| 2  | 23.2   | 242    | 1      |   |   |   |
| 3  | 24     | 290    | 1      |   |   |   |
| 4  | 23.9   | 340    | 1      |   |   |   |
| 5  | 26.3   | 363    | 1      |   |   |   |
| 6  | 26.5   | 430    | 1      |   |   |   |
| 7  | 26.8   | 450    | 1      |   |   |   |
| 8  | 26.8   | 500    | 1      |   |   |   |
| 9  | 27.6   | 390    | 1      |   |   |   |
| 10 | 27.6   | 450    | 1      |   |   |   |
| 11 | 28.5   | 500    | 1      |   |   |   |
| 12 | 28.4   | 475    | 1      |   |   |   |
| 13 | 28.7   | 500    | 1      |   |   |   |
| 14 | 29.1   | 500    | 1      |   |   |   |
| 15 | 29.5   | 340    | 1      |   |   |   |
| 16 | 29.4   | 600    | 1      |   |   |   |

## 10> 머신러닝 실행

### \*컨셉

|    | A      | B      | C | D | E | F |
|----|--------|--------|---|---|---|---|
| 1  | Length | Weight |   |   |   |   |
| 2  | 23.2   | 242    |   |   |   |   |
| 3  | 24     | 290    |   |   |   |   |
| 4  | 23.9   | 340    |   |   |   |   |
| 5  | 26.3   | 363    |   |   |   |   |
| 6  | 26.5   | 430    |   |   |   |   |
| 7  | 26.8   | 450    |   |   |   |   |
| 8  | 26.8   | 500    |   |   |   |   |
| 9  | 27.6   | 390    |   |   |   |   |
| 10 | 27.6   | 450    |   |   |   |   |
| 11 | 28.5   | 500    |   |   |   |   |
| 12 | 28.4   | 475    |   |   |   |   |
| 13 | 28.7   | 500    |   |   |   |   |
| 14 | 29.1   | 500    |   |   |   |   |
| 15 | 29.5   | 340    |   |   |   |   |
| 16 | 29.4   | 600    |   |   |   |   |





## 10> 머신러닝 실행

### \*컨셉

|    | A      | B      | C | D | E | F |
|----|--------|--------|---|---|---|---|
| 1  | Length | Weight |   |   |   |   |
| 2  | 23.2   | 242    | 1 |   |   |   |
| 3  | 24     | 290    | 1 |   |   |   |
| 4  | 23.9   | 340    | 1 |   |   |   |
| 5  | 26.3   | 363    | 1 |   |   |   |
| 6  | 26.5   | 430    | 1 |   |   |   |
| 7  | 26.8   | 450    | 1 |   |   |   |
| 8  | 26.8   | 500    | 1 |   |   |   |
| 9  | 27.6   | 390    | 1 |   |   |   |
| 10 | 27.6   | 450    | 1 |   |   |   |
| 11 | 28.5   | 500    | 1 |   |   |   |
| 12 | 28.4   | 475    | 1 |   |   |   |
| 13 | 28.7   | 500    | 1 |   |   |   |
| 14 | 29.1   | 500    | 1 |   |   |   |
| 15 | 29.5   | 340    | 1 |   |   |   |
| 16 | 29.4   | 600    | 1 |   |   |   |



## 10> 머신러닝 실행

### \*컨셉

|    | A      | B      | C      | D | E | F |
|----|--------|--------|--------|---|---|---|
| 1  | Length | Weight | target |   |   |   |
| 2  | 23.2   | 242    | 1      |   |   | 1 |
| 3  | 24     | 290    | 1      |   |   | 1 |
| 4  | 23.9   | 340    | 1      |   |   | 1 |
| 5  | 26.3   | 363    | 1      |   |   | 1 |
| 6  | 26.5   | 430    | 1      |   |   | 1 |
| 7  | 26.8   | 450    | 1      |   |   | 1 |
| 8  | 26.8   | 500    | 1      |   |   | 1 |
| 9  | 27.6   | 390    | 1      |   |   | 1 |
| 10 | 27.6   | 450    | 1      |   |   | 1 |
| 11 | 28.5   | 500    | 1      |   |   | 1 |
| 12 | 28.4   | 475    | 1      |   |   | 1 |
| 13 | 28.7   | 500    | 1      |   |   | 1 |
| 14 | 29.1   | 500    | 1      |   |   | 1 |
| 15 | 29.5   | 340    | 1      |   |   | 1 |
| 16 | 29.4   | 600    | 1      |   |   | 1 |



## 10> 머신러닝 실행

```
from sklearn.neighbors import KNeighborsClassifier  
#from sklearn이라는 패키지에서 KNeighborsClassifier 사용
```

```
Kn=KNeighborsClassifier()  
#훈련을 하기 위한 객체 생성
```

```
kn.fit(fish_data, fish_target)  
# 훈련 시작
```

```
kn.score(fish_data, fish_target)  
# 평가
```

## 10> 머신러닝 실행

```
[45] from sklearn.neighbors import KNeighborsClassifier
```

```
[46] kn=KNeighborsClassifier()
```

```
[47] kn.fit(fish_data, fish_target)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,  
                     weights='uniform')
```

```
[48] kn.score(fish_data, fish_target)
```

```
1.0
```

## 11> 맞춰보기

길이가 30cm이고 무게가 600g이면 무엇인가요?

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

#길이 30cm, 무게 600g 이면 무엇인가요?

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

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

## 11> 머신러닝의 평가

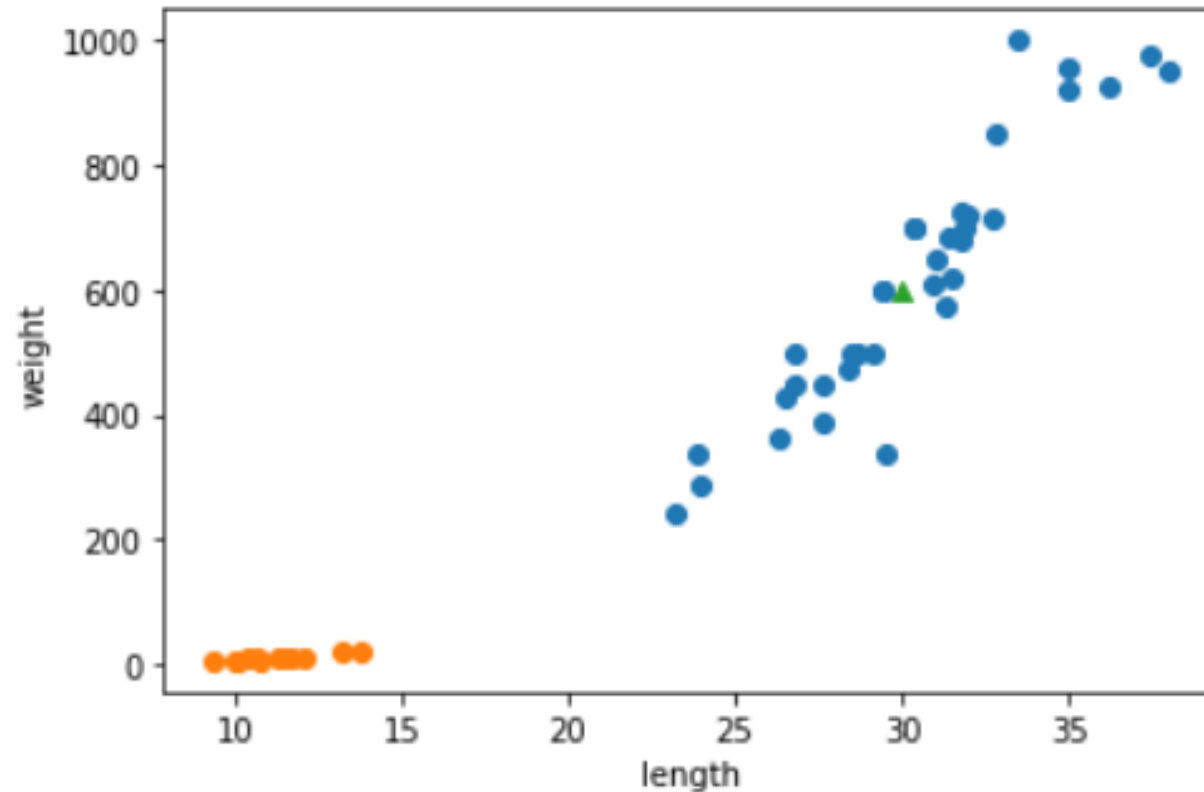
: 도미(bream), 빙어(smelt)를 시각화 하고, 길이30cm  
무게600g인 물고기를 표시

```
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()
```

## 11> 머신러닝의 평가

: 도미(bream), 빙어(smelt)를 시각화 하고, 길이30cm

무게600g인 물고기를 표시





## Chapter03 : Q&A

