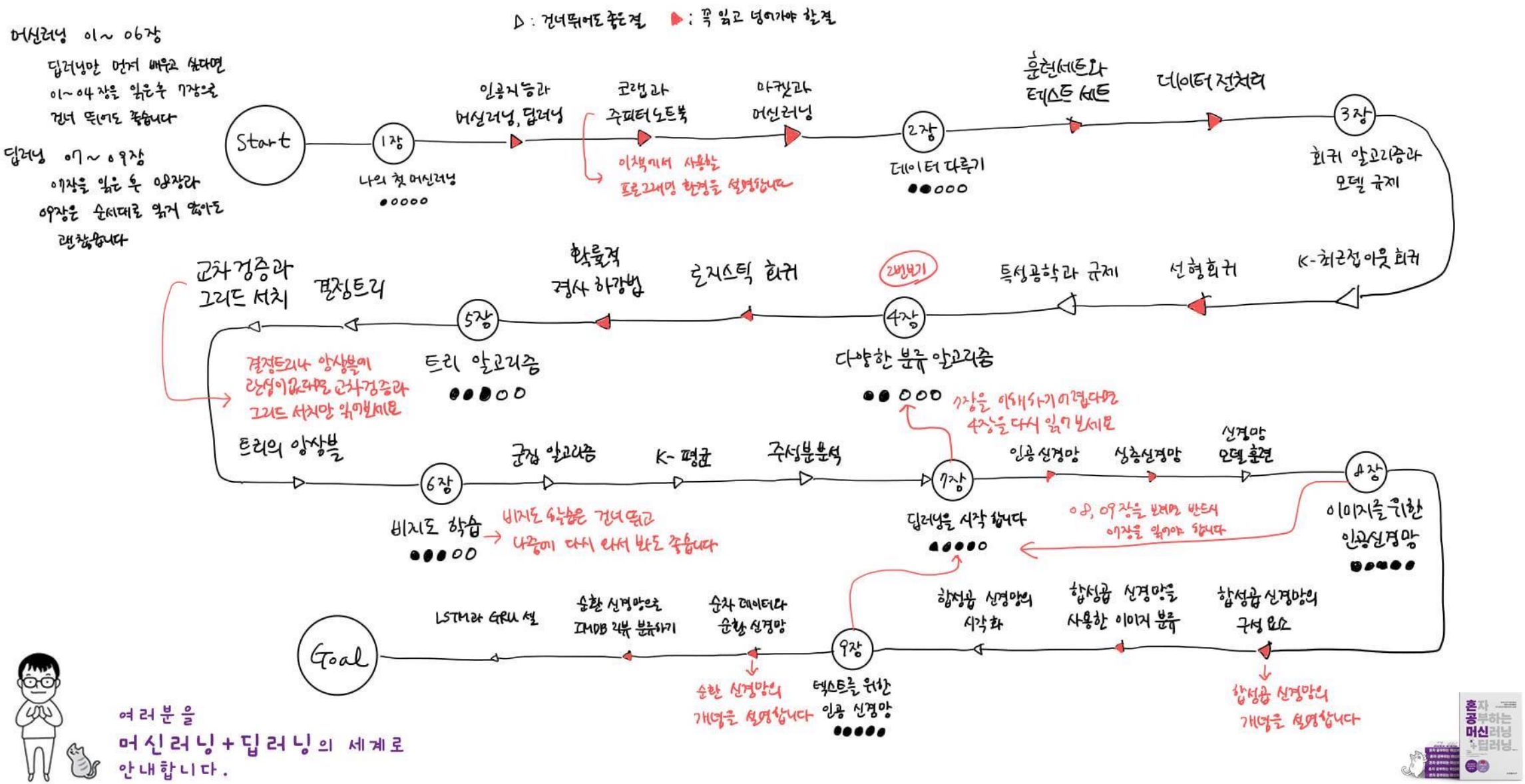
혼자 공부하는 머신러닝 + 딥러닝

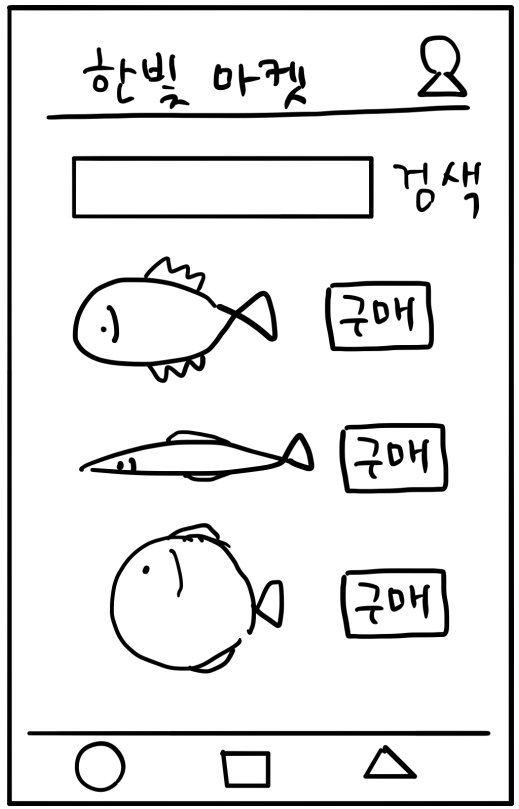
01-3 마켓과 머신러닝

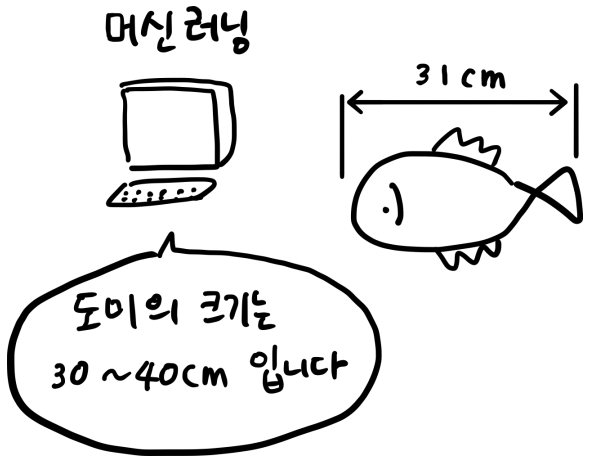
박해선



여러분을
머신러닝+딥러닝의 세계로
안내합니다.

첫 번째 머신러닝 프로그램



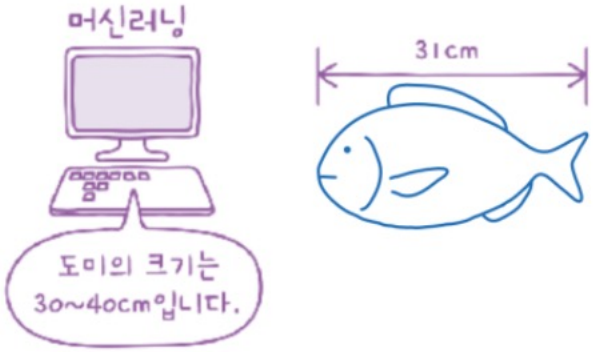


한빛 마켓은 앱 마켓 최초로 살아 있는 생선을 판매하기 시작했습니다. 고객이 온라인으로 주문하면 가장 빠른 물류 센터에서 신선한 생선을 곧바로 배송합니다. 그런데 한 가지 문제가 생겼습니다. 물류 센터에서 생선을 고르는 직원이 도통 생선 이름을 외우지 못하는 것입니다. 항상 주위 사람에게 "이 생선 이름이 뭐예요?"라고 물어봐 배송이 지연되기 일쑤입니다. 이 소식을 들은 김 팀장은 혼공머신에게 첫 번째 임무로 생선 이름을 자동으로 알려주는 머신러닝을 만들라고 맡겼습니다. 그럼 혼공머신과 함께 이 문제를 해결해 볼까요?

전통적인 프로그램

생선 분류 문제

한빛 마켓에서 팔기 시작한 생선은 '도미', '곤들매기', '농어', '강꼬치고기', '로치', '빙어', '송어'입니다. 이 생선들은 물류 센터에 많이 준비되어 있습니다. 이 생선들을 프로그램으로 분류한다고 가정해 봅시다. 어떻게 프로그램을 만들어야 할까요?



✚ 여기서 잠깐 **생선 데이터셋의 출처**

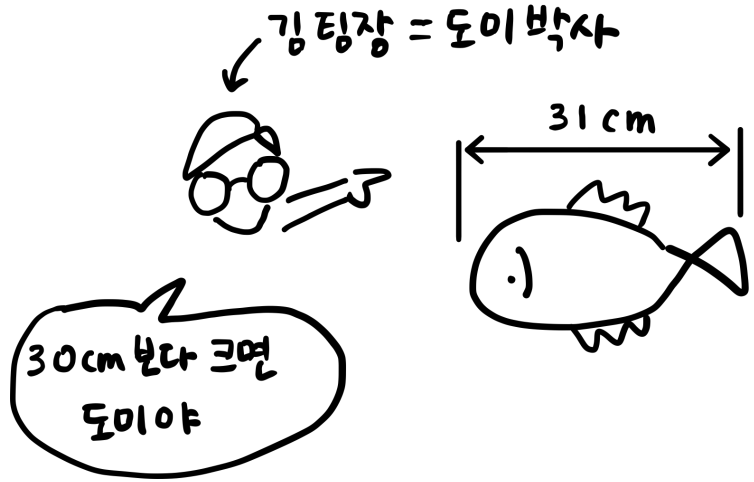
이번에 사용할 생선 데이터는 캐글에 공개된 데이터셋입니다.

- https://www.kaggle.com/aungpyaeap/fish-market

캐글(kaggle.com)은 2010년에 설립된 전 세계에서 가장 큰 머신러닝 경연 대회 사이트입니다. 대회 정보뿐만 아니라 많은 데이터와 참고 자료를 제공합니다.

[실습]
https://colab.research.google.com/drive/1MXuHcXWJakWbTxlmySjk2S1k44O4gHtf

아무래도 생선을 분류하는 일이니 생선의 특징을 알면 쉽게 구분할 수 있을 것 같습니다. 마침 김 팀장이 도미에 대해 잘 안다며 혼공머신에게 생선 길이가 30cm 이상이면 도미라고 알려줬습니다. 그 이야기를 듣고 혼공머신은 다음과 같은 파이썬 프로그램을 만들었습니다.




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

- 도미 vs 도미가 아닌 생선
- 도미 vs 빙어

도미 vs 빙어

2개의 클래스(class)

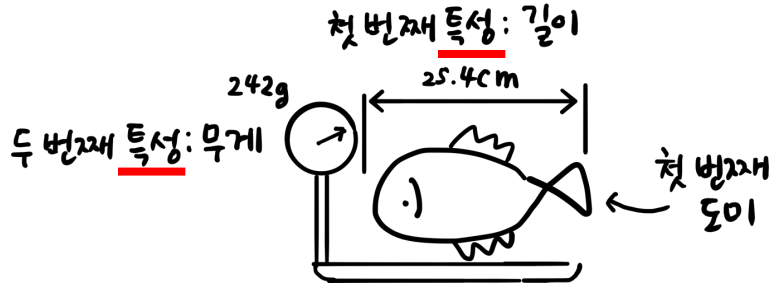
분류(classification)

이진 분류(binary classification)

✚ 여기서 잠깐 **이진 분류**

머신러닝에서 여러 개의 종류(혹은 **클래스**(class)라고 부릅니다) 중 하나를 구별해 내는 문제를 **분류**(classification)라고 부릅니다. 특히 이 장에서처럼 2개의 클래스 중 하나를 고르는 문제를 **이진 분류**(binary classification)라고 합니다. 여기

도미 데이터 – 35마리



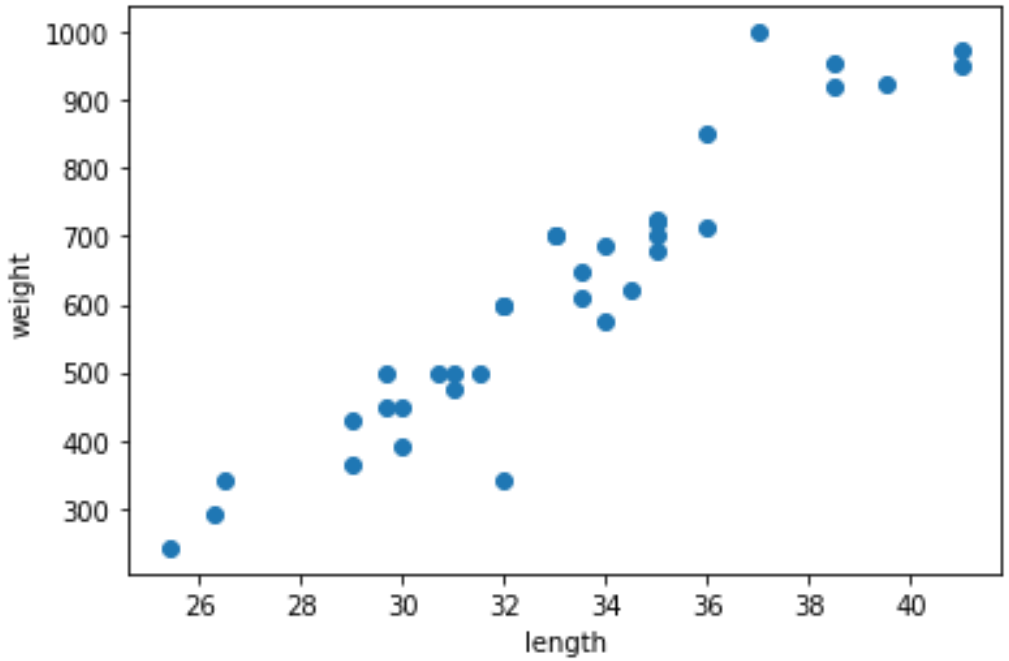

```
bream_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7,
                31.0, 31.0, 31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5,
                34.0, 34.0, 34.5, 35.0, 35.0, 35.0, 35.0, 36.0, 36.0, 37.0,
                38.5, 38.5, 39.5, 41.0, 41.0]
bream_weight = [242.0, 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]
```

[핵심키워드: 특성(feature)]

산점도(scatter plot)

```
import matplotlib.pyplot as plt

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



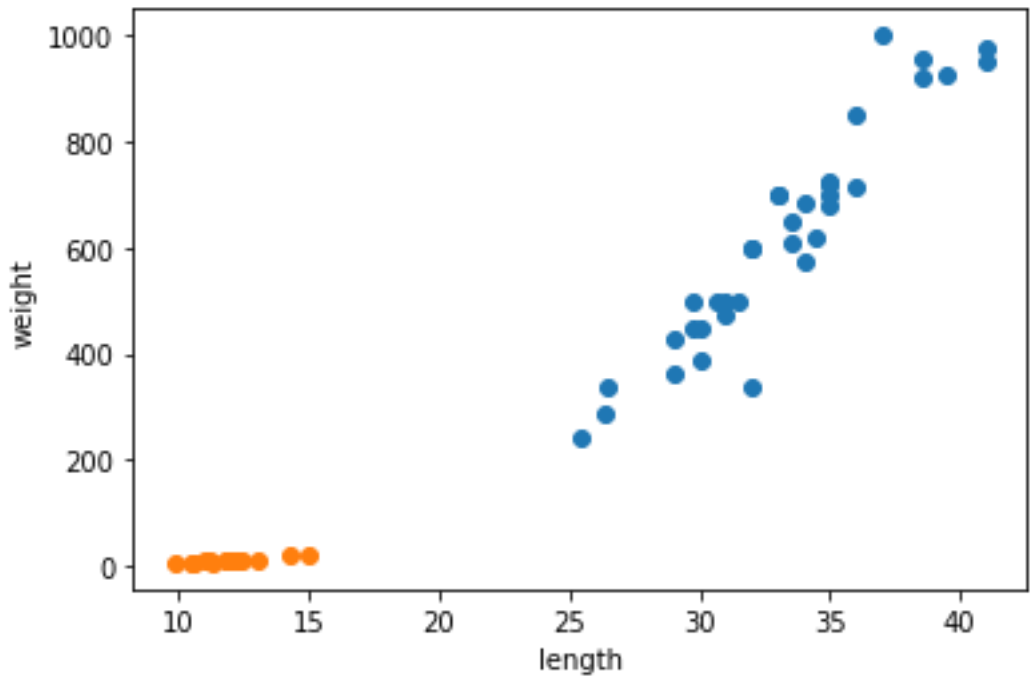
두 특성을 숫자로 보는 것보다 그래프로 표현하면 데이터를 잘 이해할 수 있고 앞으로 할 작업에 대한 힌트를 얻을 수도 있습니다. 길이를 x축으로 하고 무게를 y축으로 정하겠습니다. 그다음 각 도미를 이 그래프에 점으로 표시해 보죠. 이런 그래프를 **산점도**scatter plot라고 부릅니다.

산점도는 x, y축으로 이뤄진 좌표계에 두 변수(x, y)의 관계를 표현하는 방법입니다.

빙어 데이터-14마리

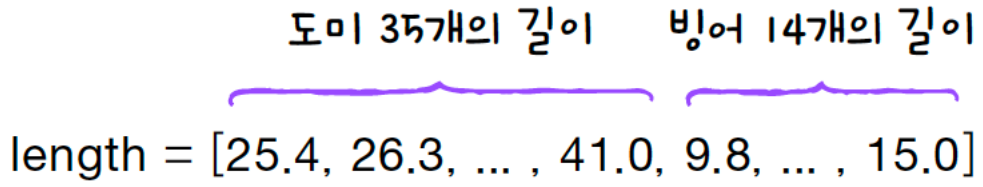
```
smelt_length = [9.8, 10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2,
                12.4, 13.0, 14.3, 15.0]
smelt_weight = [6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4,
                12.2, 19.7, 19.9]
```

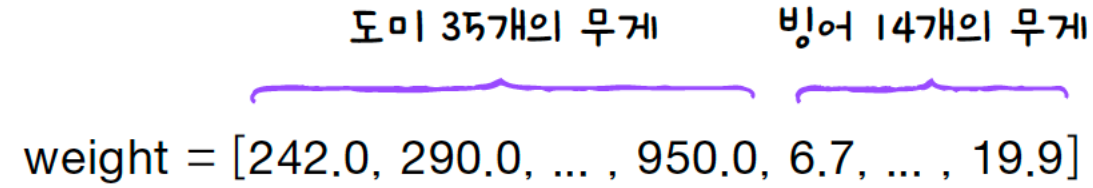
```
plt.scatter(bream_length, bream_weight)
plt.scatter(smelt_length, smelt_weight)
plt.xlabel('length')
plt.ylabel('weight')
plt.show()
```



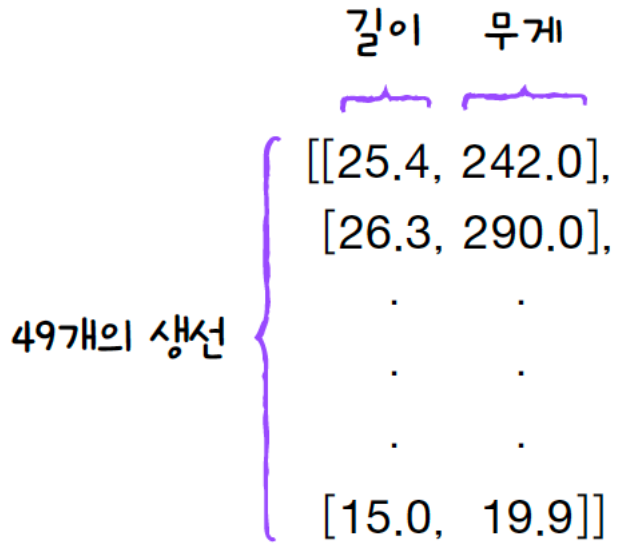
도미와 빙어 합치기

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





사이킷런이 기대하는 데이터 형태



리스트 내포

```
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]]
```

정답 준비

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

```
[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]
```

KNeighborsClassifier

[document]

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[source]

https://github.com/scikit-learn/scikit-learn/blob/70fdc843a4b8182d97a3508c1a426acc5e87e980/sklearn/neighbors/_classification.py#L238

접 이웃입니다. 이 알고리즘에 대해 조금 더 자세히 알아보도록 하겠습니다. k-최근접 이웃 알고리즘은 매우 간단합니다. 어떤 데이터에 대한 답을 구할 때 주위의 다른 데이터를 보고 다수를 차지하는 것을 정답으로 사용합니다. 마치 근묵자흑과 같이 주위의 데이터로 현재 데이터를 판단하는 거죠.

k-최근접 이웃

```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier()

kn.fit(fish_data, fish_target)

kn.score(fish_data, fish_target)
```
```
1.0
```

[참고] 머신러닝 용어

이 객체에 fish_data와 fish_target을 전달하여 도미를 찾기 위한 기준을 학습시킵니다. 이런 과정을 머신러닝에서는 **훈련**training이라고 부릅니다. 사이킷런에서는 fit() 메서드가 이런 역할을 합니다. 이 메서드에 fish_data와 fish_target을 순서대로 전달해 보겠습니다.

모델에 데이터를 전달하여 규칙을 학습하는 과정을 훈련이라고 합니다.

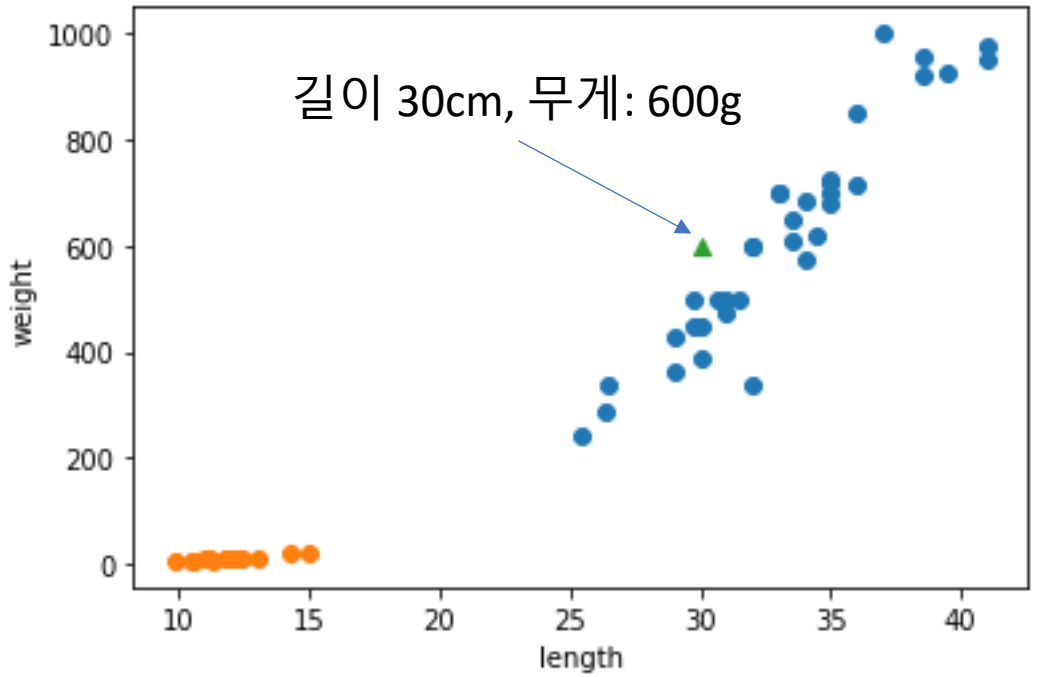
✚ 여기서 잠깐 머신러닝에서의 모델

머신러닝 알고리즘을 구현한 프로그램을 **모델**(model)이라고 부릅니다. 또는 프로그램이 아니더라도 알고리즘을 (수식 등으로) 구체화하여 표현한 것을 모델이라고 부릅니다. 예를 들어 "스팸 메일을 걸러내기 위해 k-최근접 이웃 모델을 사용해 봅시다"라고 말할 수 있습니다.

이제 객체(또는 모델) kn이 얼마나 잘 훈련되었는지 평가해 보겠습니다. 사이킷런에서 모델을 평가하는 메서드는 score() 메서드입니다. 이 메서드는 0에서 1 사이의 값을 반환합니다. 1은 모든 데이터를 정확히 맞혔다는 것을 나타냅니다. 예를 들어 0.5라면 절반만 맞혔다는 의미죠.

새로운 생선 예측

예를 들어 다음 그림에 삼각형으로 표시된 새로운 데이터가 있다고 가정해 보죠. 이 삼각형은 도미와 빙어 중 어디에 속할까요?




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

array([1])
```

K-NN의 단점

이렇게 생각하면 k-최근접 이웃 알고리즘을 위해 준비해야 할 일은 데이터를 모두 가지고 있는 게 전부입니다. 새로운 데이터에 대해 예측할 때는 가장 가까운 직선거리에 어떤 데이터가 있는지를 살피기만 하면 됩니다. 단점은 k-최근접 이웃 알고리즘의 이런 특징 때문에 데이터가 아주 많은 경우 사용하기 어렵습니다. 데이터가 크기 때문에 메모리가 많이 필요하고 직선거리를 계산하는 데도 많은 시간이 필요합니다.

그럼 가까운 몇 개의 데이터를 참고할까요? 이는 정하기 나름입니다. KNeighborsClassifier 클래스의 기본값은 5입니다. 이 기준은 n_neighbors 매개변수로 바꿀 수 있습니다. 예를 들어 다음과 같이 하면 어떤 결과가 나올까요?

손코딩

```
kn49 = KNeighborsClassifier(n_neighbors=49)  # 참고 데이터를 49개로 한 kn49 모델
```

무조건 도미

```
kn49 = KNeighborsClassifier(n_neighbors=49)

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

```
0.7142857142857143
```

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

```
0.7142857142857143
```

[정리]

도미와 빙어 분류 문제해결 과정

혼공머신은 도미와 빙어를 구분하기 위해 첫 번째 머신러닝 프로그램을 만들었습니다. 먼저 도미 35마리와 빙어 14마리의 길이와 무게를 측정해서 파이썬 리스트로 만듭니다. 그다음 도미와 빙어 데이터를 합쳐 리스트의 리스트로 데이터를 준비했습니다.

혼공머신이 사용한 첫 번째 머신러닝 알고리즘은 k-최근접 이웃 알고리즘입니다. 사이킷런의 k-최근접 이웃 알고리즘은 주변에서 가장 가까운 5개의 데이터를 보고 다수결의 원칙에 따라 데이터를 예측합니다. 이 모델은 혼공머신이 준비한 도미와 빙어 데이터를 모두 완벽하게 맞혔습니다.

도미와 빙어를 분류하는 문제를 풀면서 KNeighborsClassifier 클래스의 fit(), score(), predict() 메서드를 사용해 보았습니다. 끝으로 k-최근접 이웃 알고리즘의 특징을 알아보았습니다.

〈문제해결 과정〉을 읽으면서 어떻게 혼공머신이 문제를 해결했는지 함께 되짚어 보세요.

[마무리]

▶ 키워드로 끝내는 핵심 포인트

- **특성**은 데이터를 표현하는 하나의 성질입니다. 이 절에서 생선 데이터 각각을 길이와 무게 특성으로 나타냈습니다.
- 머신러닝 알고리즘이 데이터에서 규칙을 찾는 과정을 **훈련**이라고 합니다. 사이킷런에서는 fit() 메서드가 하는 역할입니다.
- **k-최근접 이웃 알고리즘**은 가장 간단한 머신러닝 알고리즘 중 하나입니다. 사실 어떤 규칙을 찾기보다는 전체 데이터를 메모리에 가지고 있는 것이 전부입니다.
- 머신러닝 프로그램에서는 알고리즘이 구현된 객체를 **모델**이라고 부릅니다. 종종 알고리즘 자체를 모델이라고 부르기도 합니다.
- **정확도**는 정확한 답을 몇 개 맞혔는지를 백분율로 나타낸 값입니다. 사이킷런에서는 0~1 사이의 값으로 출력됩니다.
  정확도 = (정확히 맞힌 개수) / (전체 데이터 개수)

[마무리]

scikit-learn

- KNeighborsClassifier()는 k-최근접 이웃 분류 모델을 만드는 사이킷런 클래스입니다. n_neighbors 매개변수로 이웃의 개수를 지정합니다. 기본값은 5입니다.

  p 매개변수로 거리를 재는 방법을 지정합니다. 1일 경우 맨해튼 거리(https://bit.ly/man_distance)를 사용하고, 2일 경우 유클리디안 거리(https://bit.ly/euc_distance)를 사용합니다. 기본값은 2입니다.

  n_jobs 매개변수로 사용할 CPU 코어를 지정할 수 있습니다. -1로 설정하면 모든 CPU 코어를 사용합니다. 이웃 간의 거리 계산 속도를 높일 수 있지만 fit() 메서드에는 영향이 없습니다. 기본값은 1입니다.

- fit()은 사이킷런 모델을 훈련할 때 사용하는 메서드입니다. 처음 두 매개변수로 훈련에 사용할 특성과 정답 데이터를 전달합니다.

- predict()는 사이킷런 모델을 훈련하고 예측할 때 사용하는 메서드입니다. 특성 데이터 하나만 매개변수로 받습니다.

- score()는 훈련된 사이킷런 모델의 성능을 측정합니다. 처음 두 매개변수로 특성과 정답 데이터를 전달합니다. 이 메서드는 먼저 predict() 메서드로 예측을 수행한 다음 분류 모델일 경우 정답과 비교하여 올바르게 예측한 개수의 비율을 반환합니다.

[확인문제]

▶ 확인 문제

1. 데이터를 표현하는 하나의 성질로써, 예를 들어 국가 데이터의 경우 인구 수, GDP, 면적 등이 하나의 국가를 나타냅니다. 머신러닝에서 이런 성질을 무엇이라고 부르나요?

① 특성

② 특질

③ 개성

④ 요소

2. 가장 가까운 이웃을 참고하여 정답을 예측하는 알고리즘이 구현된 사이킷런 클래스는 무엇인가요?

① SGDClassifier

② LinearRegression

③ RandomForestClassifier

④ KNeighborsClassifier

3. 사이킷런 모델을 훈련할 때 사용하는 메서드는 어떤 것인가요?

① predict()

② fit()

③ score()

④ transform()

감사합니다