

Chapter 01. Machine Learning? Human?



머신러닝이란?

머신러닝이란?

머신러닝의 교과서적 정의

?

머신러닝이란?

머신러닝은 명시적으로
프로그래밍하지 않고도 컴퓨터에
학습할 수 있는 능력을 부여하는 학문

(과거 데이터로부터 얻은)
경험이 쌓여 감에 따라 주어진 태스크의
성능이 점점 좋아질 때
컴퓨터 프로그램은 경험으로부터
학습한다고 할 수 있음

Arthur Samuel (1959)

Tom Mitchell (1998)

- 명시적인 프로그램에 의해서가 아니라, 주어진 데이터를 통해 규칙을 찾는 것
- 누가? 머신이~~ : Machine Learning
- 원리는 뭘까요?

머신러닝이란?

흔히 말하는 머신러닝(딥러닝)의 역사 1

- 아더 사缪엘(Arthur Samuel)
 - 처음으로 머신러닝(Machine Learning)이라는 용어 사용
 - 체커 게임(1959) – 스스로 학습하는 최초의 프로그램
- IBM 딥블루(Deep Blue)
 - 세계 체스 챔피언 가리 카스파로프에 승리(1997)
- IBM 왓슨(Watson)
 - TV 퀴즈 쇼 제퍼디(Jeopardy!)에서 우승(2011)
- DeepMind 알파고(AlphaGo)
 - 이세돌에게 4:1 승리(2016)
 - 2017.5.27 현재 공식 프로 대국 전적 = 13전 12승 1패
- 리브라터스(Libratus)와 딥스택(DeepStack)
 - 각각 텍사스 홀덤 포커 게임과 무제한 배팅 포커 게임에서 승리(2017)



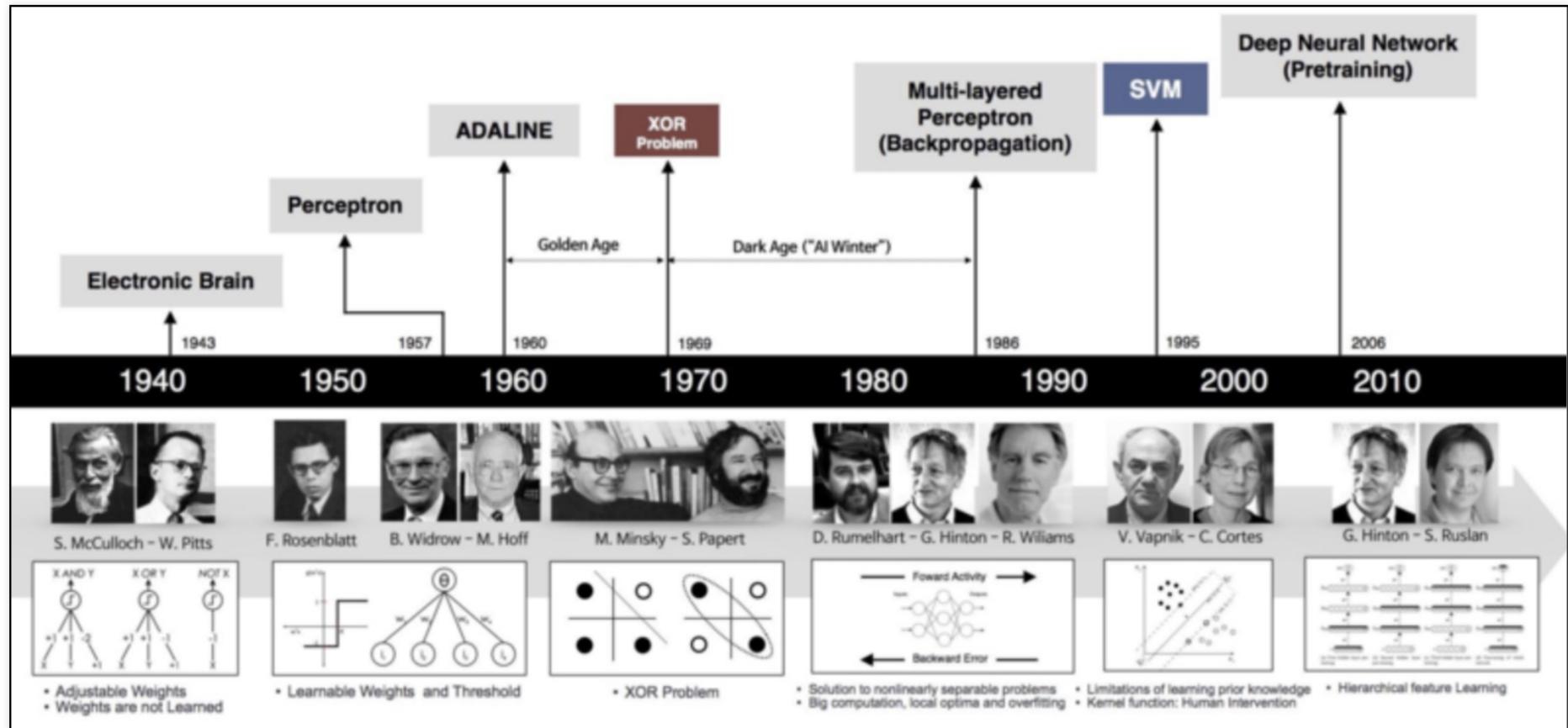
머신러닝이란?

흔히 말하는 머신러닝(딥러닝)의 역사 2

- 1950년: 앨런 튜링의 튜링 테스트 – 기계지능의 테스트 방법 제안
- 1956년: 인공지능 용어 탄생(다트머스 회의, 존 매카시)
- 1962년: 로젠블롯의 퍼셉트론(Perceptron) – 최초의 인공신경망
- 1969년: 퍼셉트론의 한계 증명(민斯基 & 페퍼트) – XOR 문제
- 1974 – 1980년 : 인공지능의 **첫번째 겨울** ← 성과 부진 + 민斯基의 비판
- 1980 – 1987년 : 인공지능의 부활(전문가 시스템, 신경망 역전파 알고리즘)
- 1987 – 1993년 : 인공지능의 **두번째 겨울** ← 신경망의 문제, 과도한 기대
- 1993년 – 현재 : 머신러닝 기법 발달, 신경망 이론의 성숙

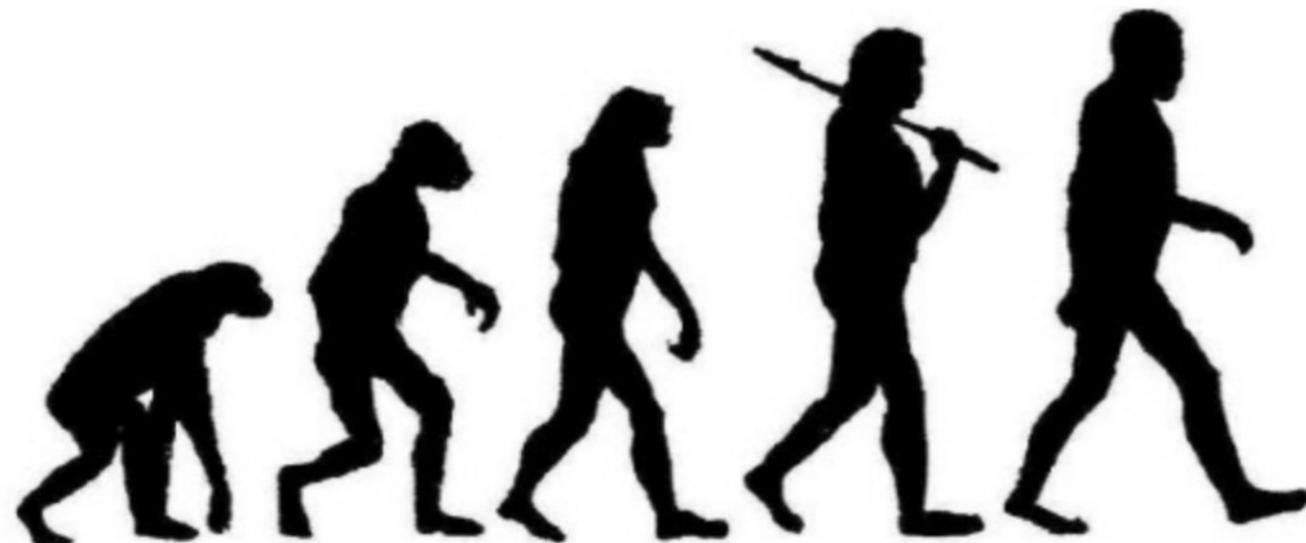
머신러닝이란?

흔히 말하는 머신러닝(딥러닝)의 역사 3



머신러닝이란?

이번에는 직접 (머신) 휴먼 러닝을 해봅시다



- 머신러닝이 어떤 것인지, 어떤 절차를 수행하는 것인지를 알기 위해
- 우리가 직접 손으로 해봅시다. (어차피 휴먼이든, 머신이든, 누구든 학습은 해야하니까)

Iris Classification

Iris Classification

IRIS



- 프랑스의 국화
- 아이리스 Iris라는 이름은 그리스 신화의 무지개의 여신인 Iris에서 따온 것.
헤라 여신이 Iris에게 내린 축복의 숨결이 땅으로 떨어져 핀 꽃
- 꽃말은 좋은 소식, 잘 전해 주세요, 사랑의 메시지

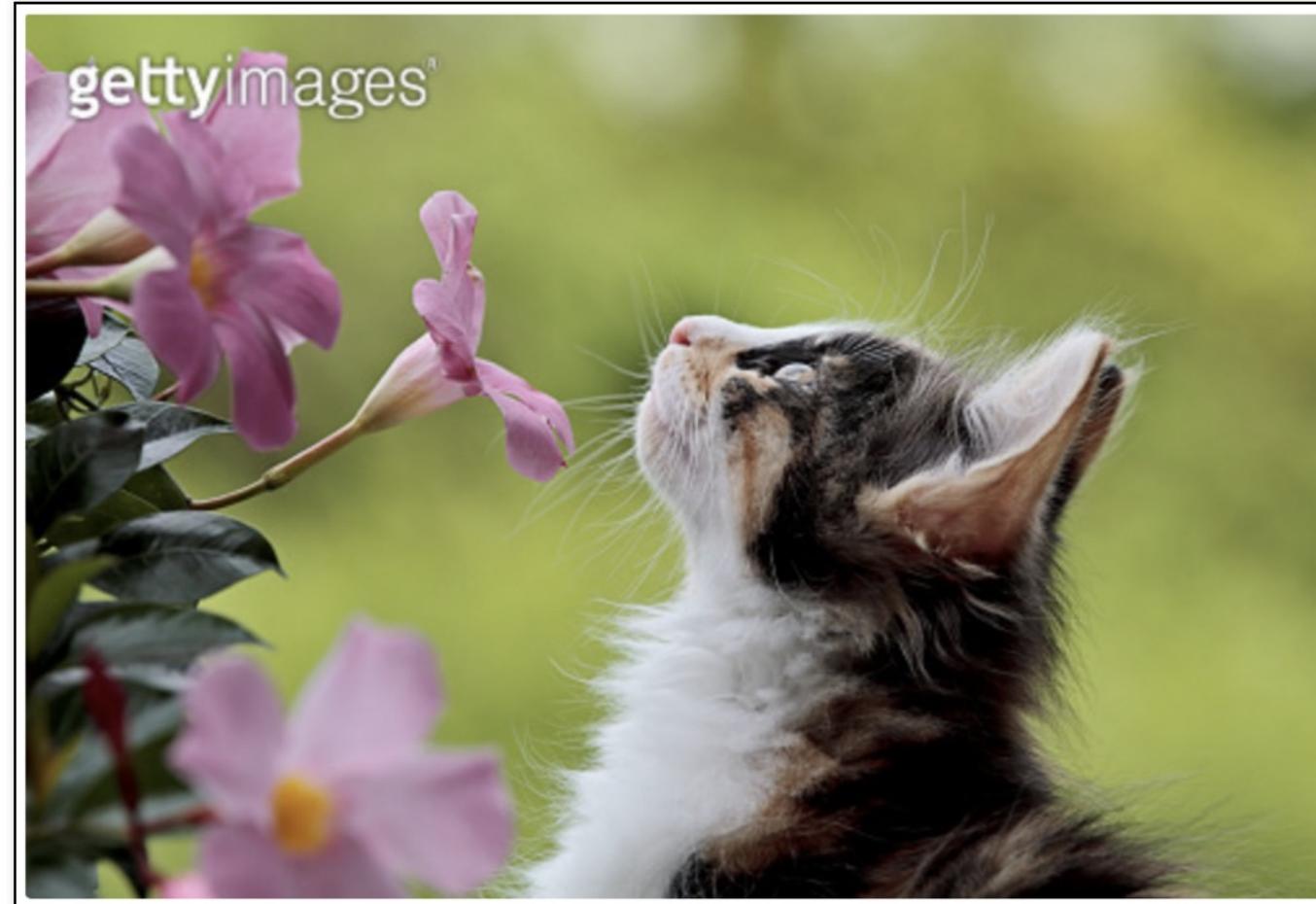
Iris Classification

Iris의 품종 분류



- 꽃잎(petal), 꽃받침(sepal)의 길이/너비 정보를 이용해서 이 3종의 품종을 구분할 수 있을까?

그러니까 어떻게 하면 IRIS를 보고,
짠~ Setosa라고 말 할 수 있을까?



일단 열심히 일해보자~



이런 생각을 한 사람이 먼저 있었다 - 로널드 피셔



- 1930년대에 통계학자이자 유전학자였던 로널드 피셔가 iris 데이터를 수집 정리해 두었다.

Iris Classification

로널드의 iris 데이터

Dataset Order ↓	Sepal length ↓	Sepal width ↓	Petal length ↓	Petal width ↓	Species ↓
1	5.1	3.5	1.4	0.2	<i>I. setosa</i>
2	4.9	3.0	1.4	0.2	<i>I. setosa</i>
3	4.7	3.2	1.3	0.2	<i>I. setosa</i>
4	4.6	3.1	1.5	0.2	<i>I. setosa</i>
5	5.0	3.6	1.4	0.3	<i>I. setosa</i>
6	5.4	3.9	1.7	0.4	<i>I. setosa</i>
7	4.6	3.4	1.4	0.3	<i>I. setosa</i>
8	5.0	3.4	1.5	0.2	<i>I. setosa</i>
9	4.4	2.9	1.4	0.2	<i>I. setosa</i>
10	4.9	3.1	1.5	0.1	<i>I. setosa</i>
11	5.4	3.7	1.5	0.2	<i>I. setosa</i>
12	4.8	3.4	1.6	0.2	<i>I. setosa</i>
13	4.8	3.0	1.4	0.1	<i>I. setosa</i>

데이터관찰 (python)

iris 데이터를 불러보자

```
| from sklearn.datasets import load_iris  
  
iris = load_iris()
```

데이터관찰 (python)

뭐가 있나?

iris

sklearn의 datasets은 Python의 dict 형과 유사하다

```
| iris.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

iris.DESCR

```
| print(iris.DESCR)

.. _iris_dataset:

Iris plants dataset
-----
**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the
else
```

target_names에는

```
| iris.target_names  
array(['setosa', 'versicolor', 'virginica'], dtype='|<U10')
```

데이터관찰 (python)

target에는

```
iris.target
```

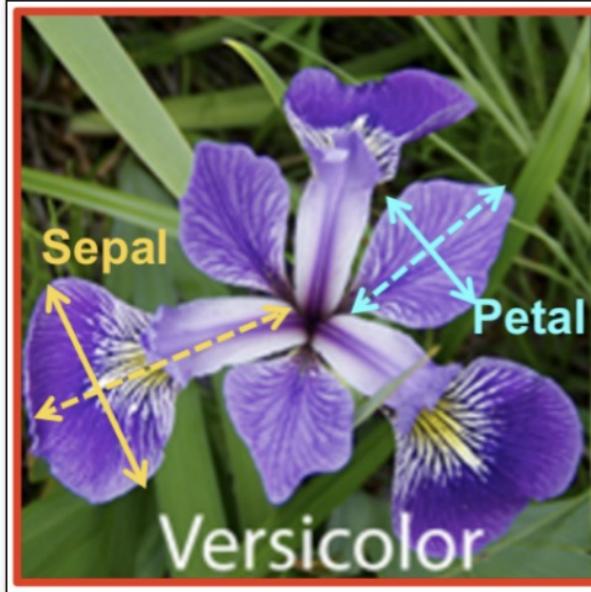
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

iris.data에는

```
| iris.data
```

```
array([[5.1, 3.5, 1.4, 0.2],  
       [4.9, 3. , 1.4, 0.2],  
       [4.7, 3.2, 1.3, 0.2],  
       [4.6, 3.1, 1.5, 0.2],  
       [5. , 3.6, 1.4, 0.2],  
       [5.4, 3.9, 1.7, 0.4],  
       [4.6, 3.4, 1.4, 0.3],  
       [5. , 3.4, 1.5, 0.2],  
       [4.4, 2.9, 1.4, 0.2],  
       [4.9, 3.1, 1.5, 0.1],
```

아무튼, petal과 sepal의 length, width로 품종을 구분할 수 있을까?



상황이라도 파악해보자. (머신러닝 말고 그냥 우리 눈으로~)



기존의 3종에 대해 먼저 공부한다
Sepal/Petal의 Width/Length에 대해서 파악

Iris 꽃의 품종을 잘 맞출 수 있는지 확인

- 그럼 어떻게 해야할까?
- 사람이 직접 구분하려면??
- 일단, 먼저 versicolor, virginica, setosa를 구분하기 위해 3개 품종의 특성을 먼저 공부해야~

데이터관찰 (python)

pandas

| 팀보이즈 베이스 gs bq

- 데이터를 바로 딥러닝에 적용하거나
- sklearn을 이용한 머신러닝에 적용할 때 꼭 필요한 것은 아니지만,
- 데이터를 정리해서 관찰할 때는 아주 유용한 도구가 pandas
- 미국식 농담으로 (pandas : 스테로이드 맞은 엑셀~)

DataFrame으로 만들어보기

```
iris_pd = pd.DataFrame(iris.data, columns=iris.feature_names)  
iris_pd.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

품종 정보도 column에 포함

```
iris_pd[ 'species' ] = iris.target  
iris_pd.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

- 이제 species에는 0, 1, 2의 값이 각 품종을 의미한다

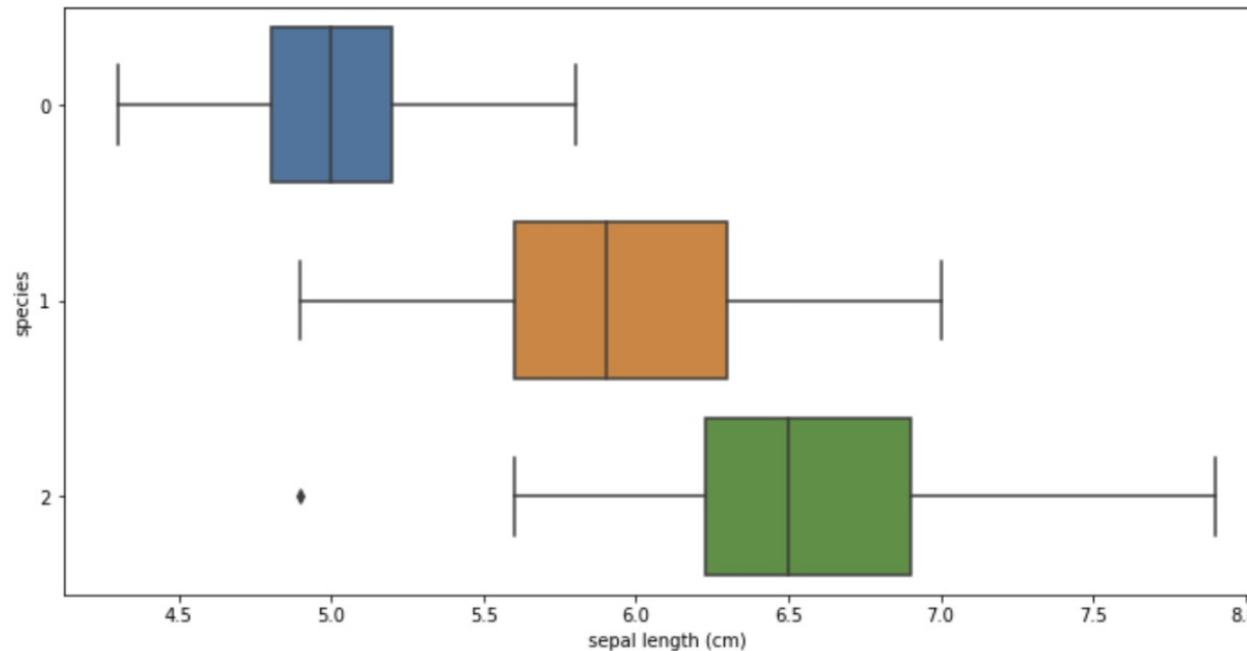
일단 그래프 그리는 모듈 import

```
| import seaborn as sns  
| import matplotlib.pyplot as plt
```

데이터관찰 (python)

boxplot(x='sepal length (cm)')

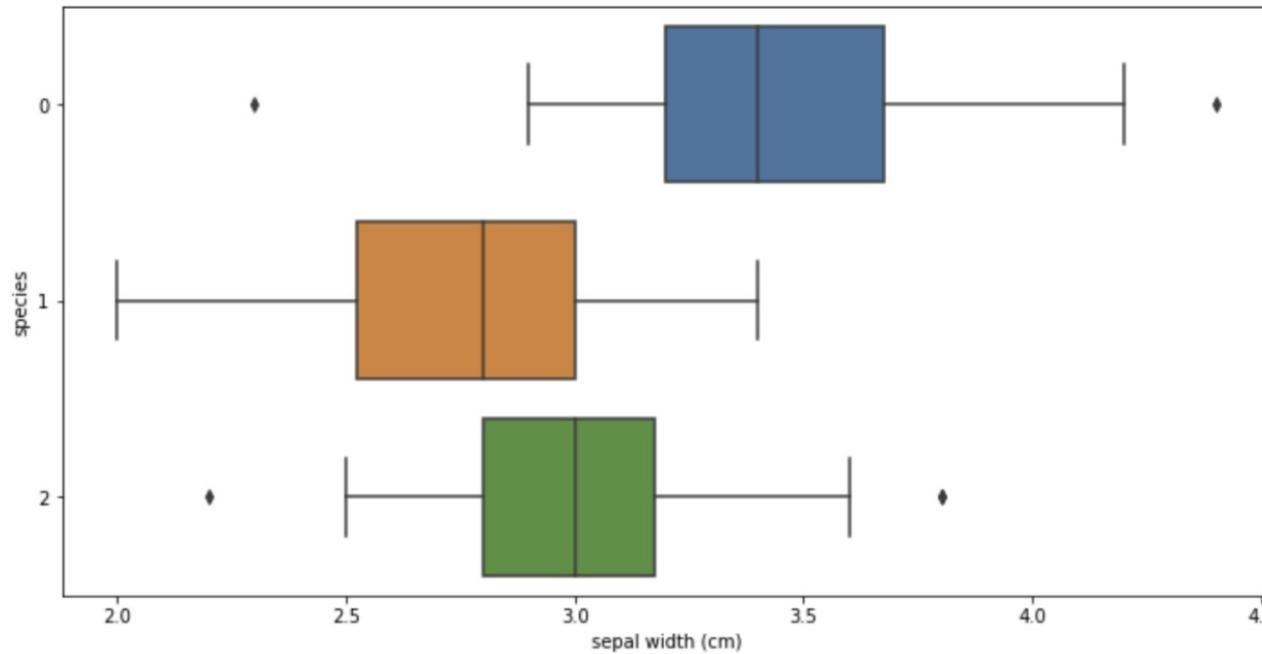
```
plt.figure(figsize=(12,6))
sns.boxplot(x='sepal length (cm)', y='species', data=iris_pd, orient='h');
```



데이터관찰 (python)

```
boxplot(x='sepal width (cm)')
```

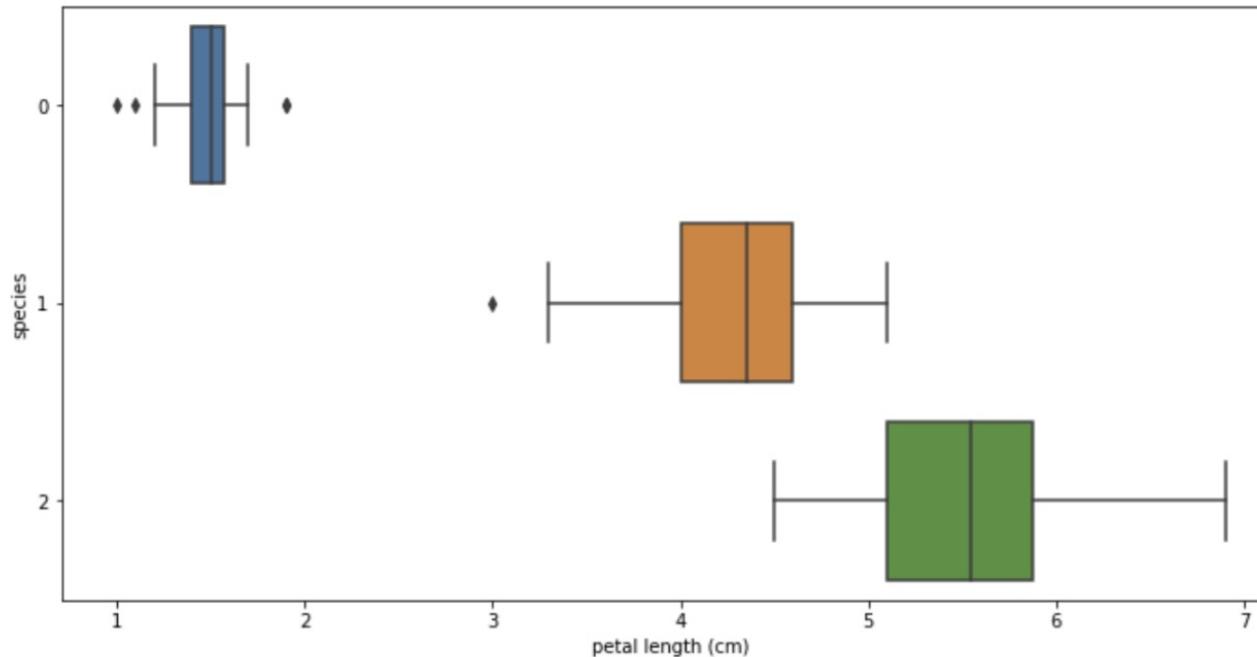
```
plt.figure(figsize=(12,6))
sns.boxplot(x='sepal width (cm)', y='species', data=iris_pd, orient='h');
```



데이터관찰 (python)

```
boxplot(x='petal length (cm)')
```

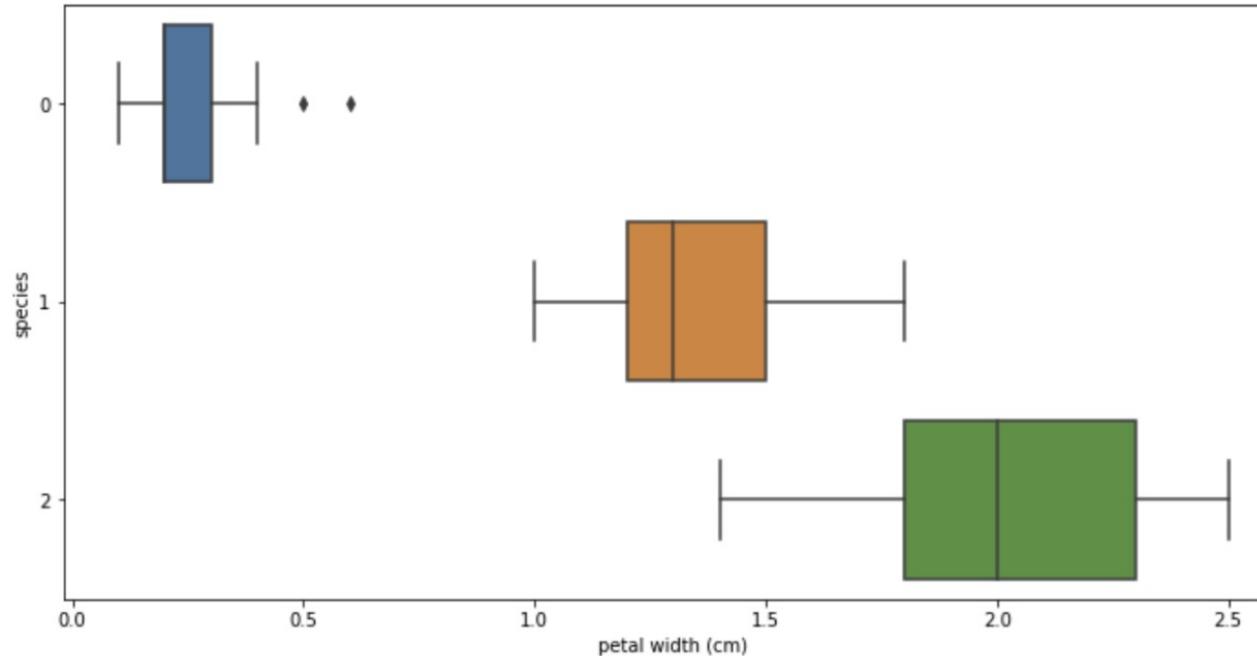
```
plt.figure(figsize=(12,6))
sns.boxplot(x='petal length (cm)', y='species', data=iris_pd, orient='h');
```



데이터관찰 (python)

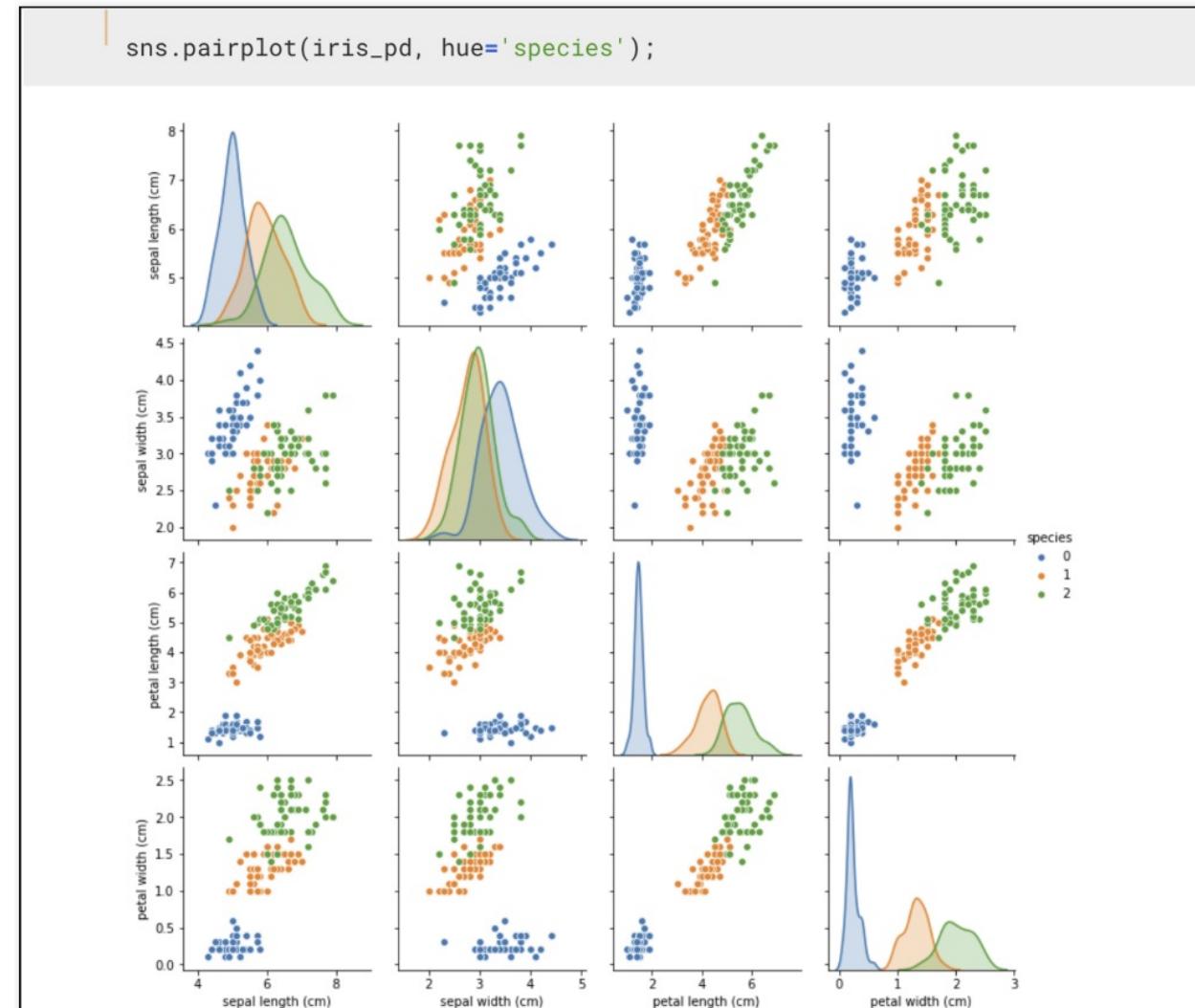
boxplot(x='petal width (cm)')

```
plt.figure(figsize=(12,6))
sns.boxplot(x='petal width (cm)', y='species', data=iris_pd, orient='h');
```



데이터관찰 (python)

pairplot

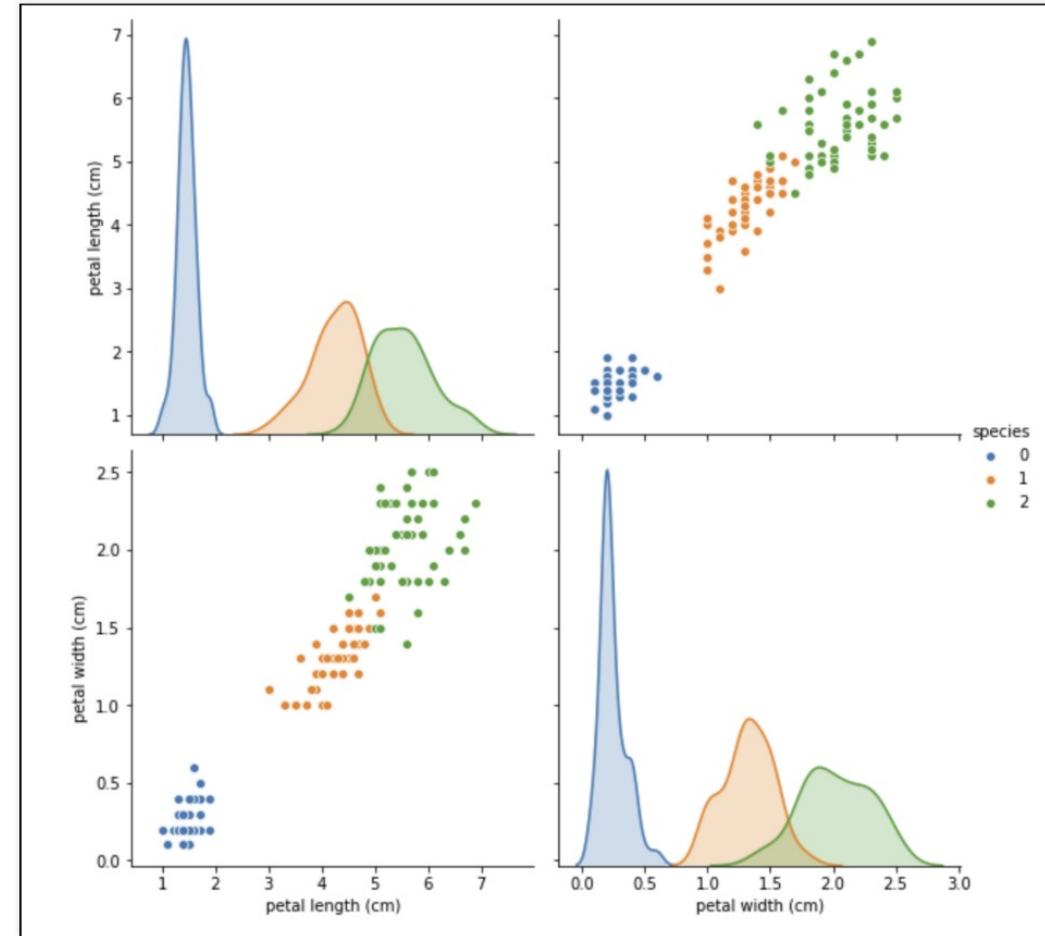


조금 더 집중적으로~

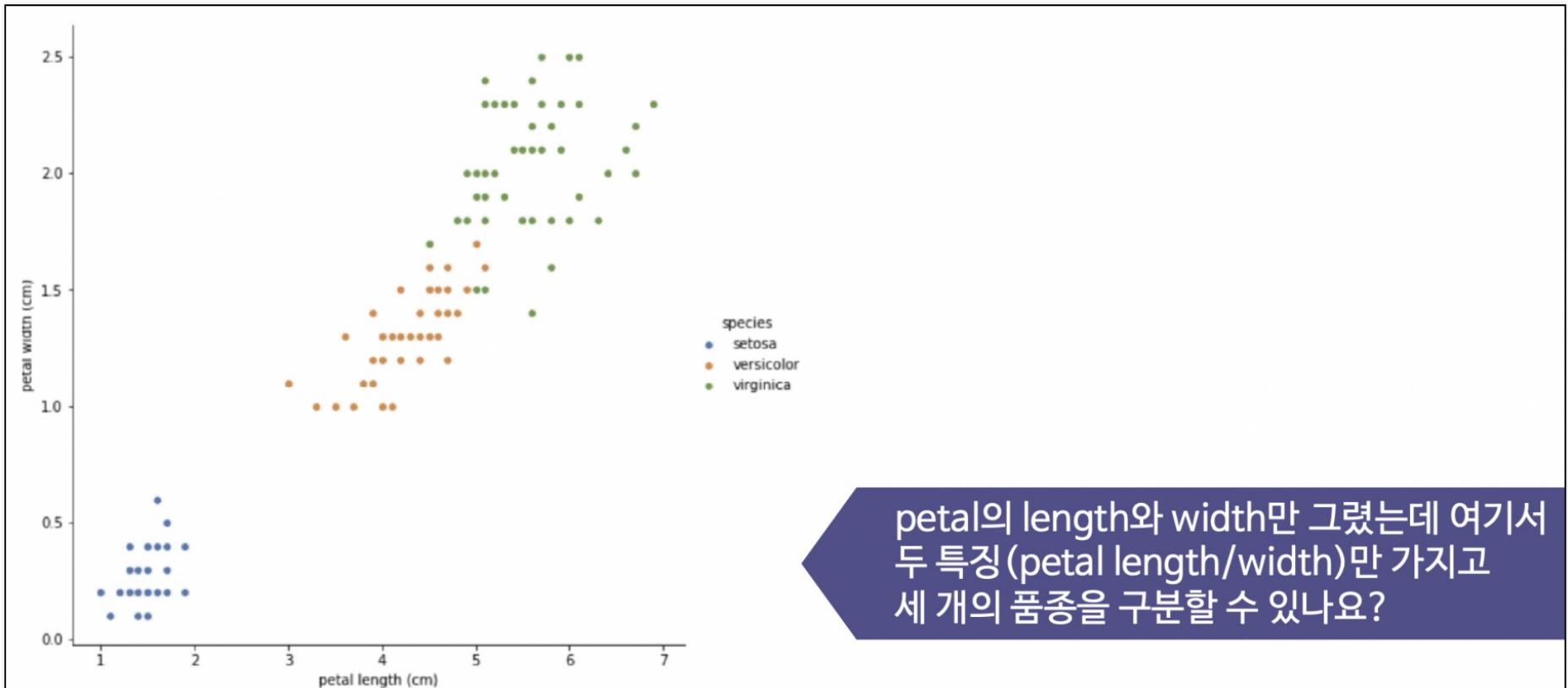
```
| sns.pairplot(data=iris_pd,  
|               vars=['petal length (cm)', 'petal width (cm)',  
|               hue='species', height=4);
```

데이터관찰 (python)

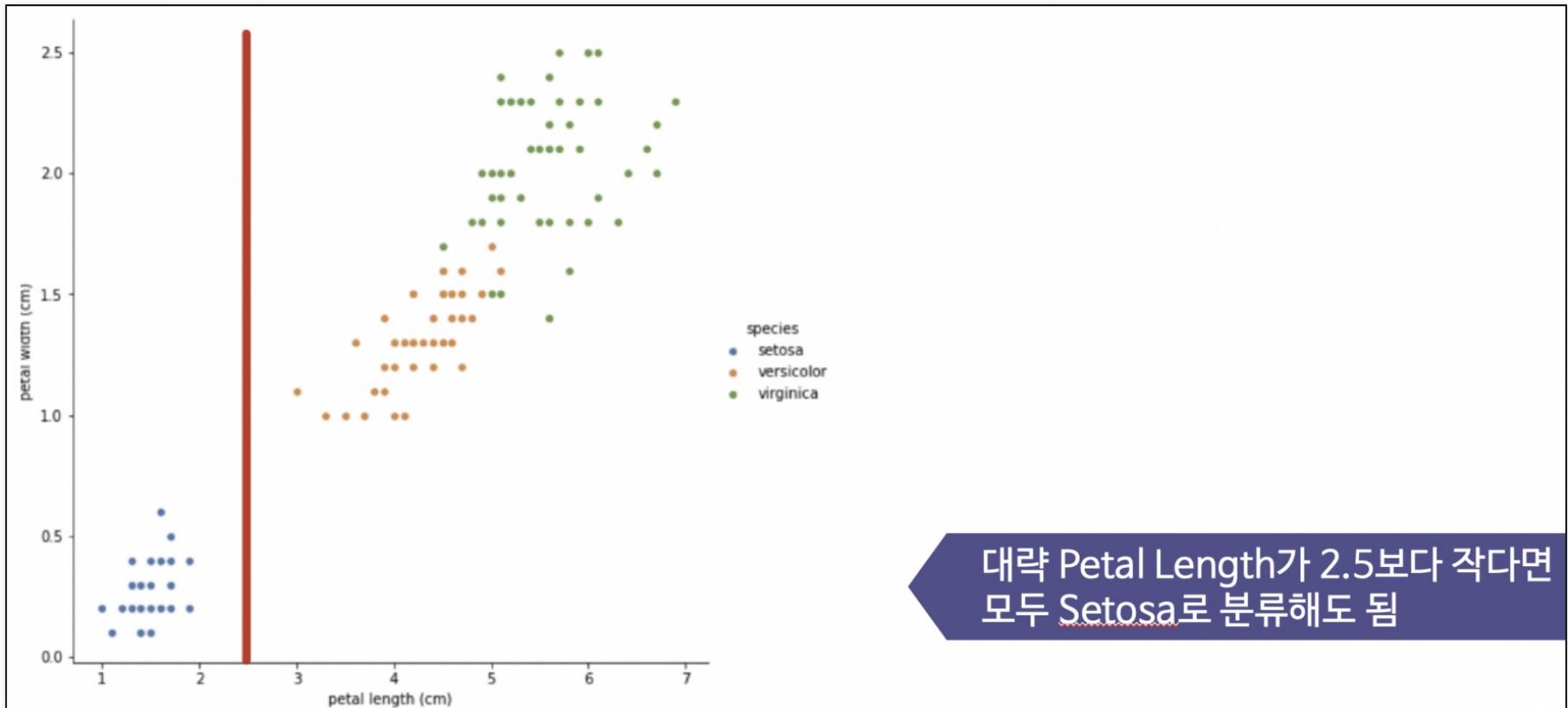
iris 3종에 대한 구분이 가능할 것 같다



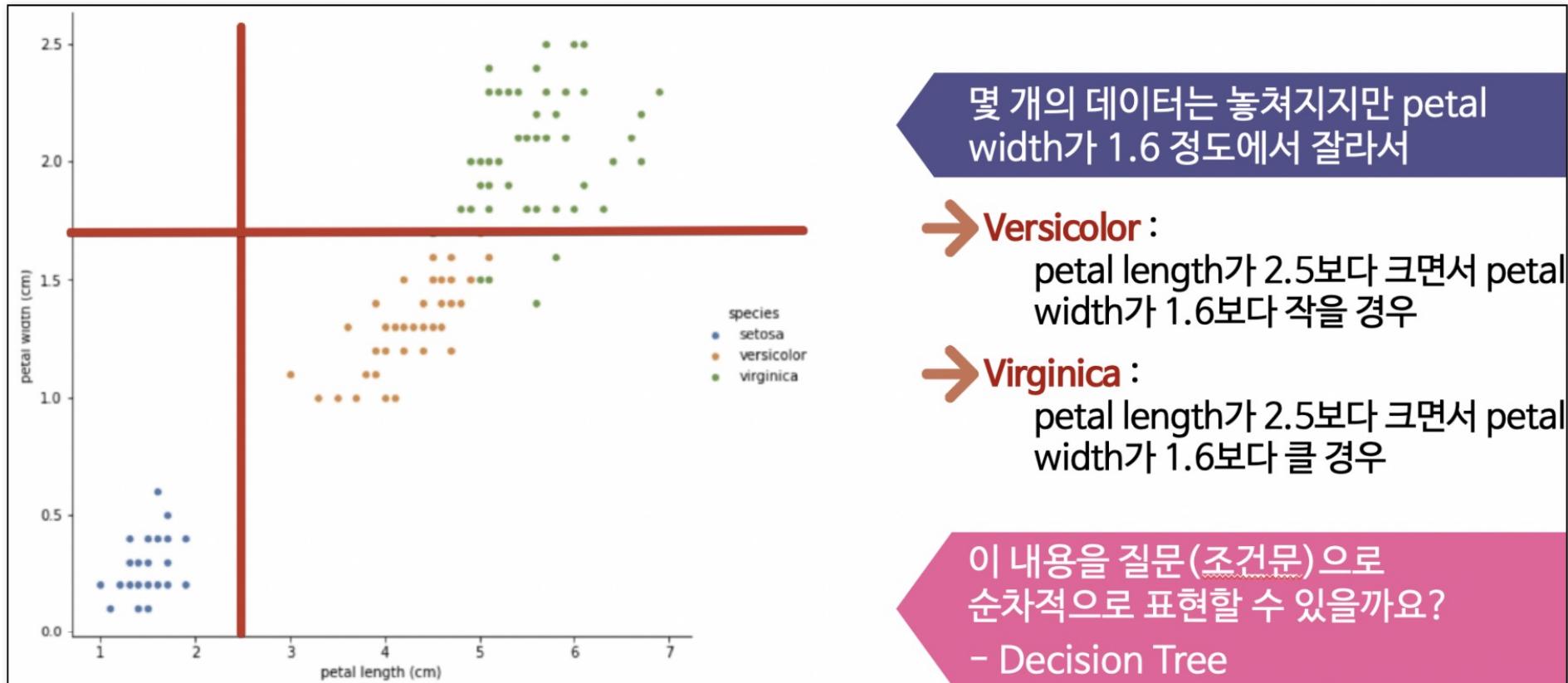
품종 세 개를 구분할 수 있을까?



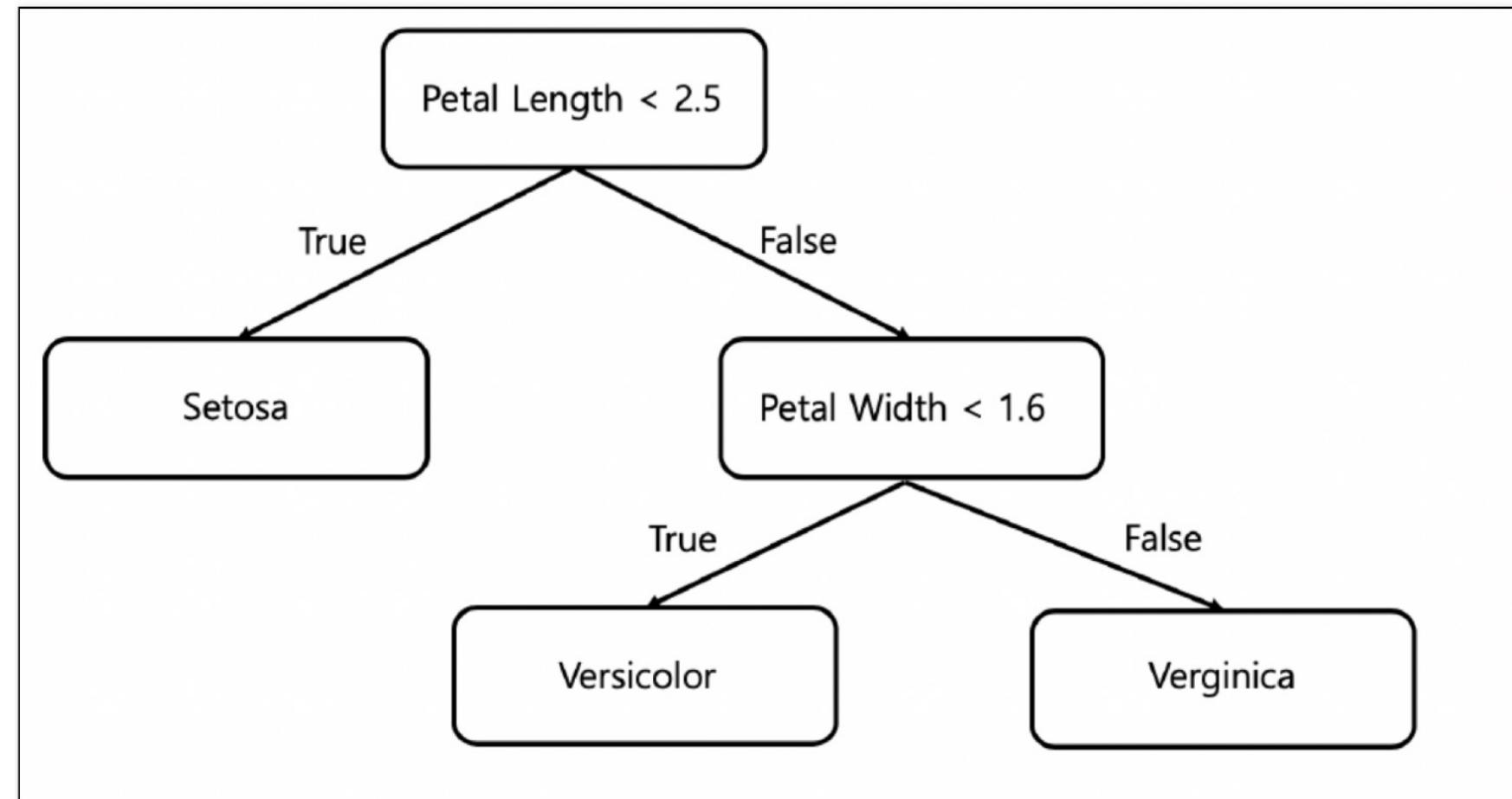
일단 setosa는 가능하다



나머지 두 개는?



Iris의 3개 품종을 구분하는 규칙



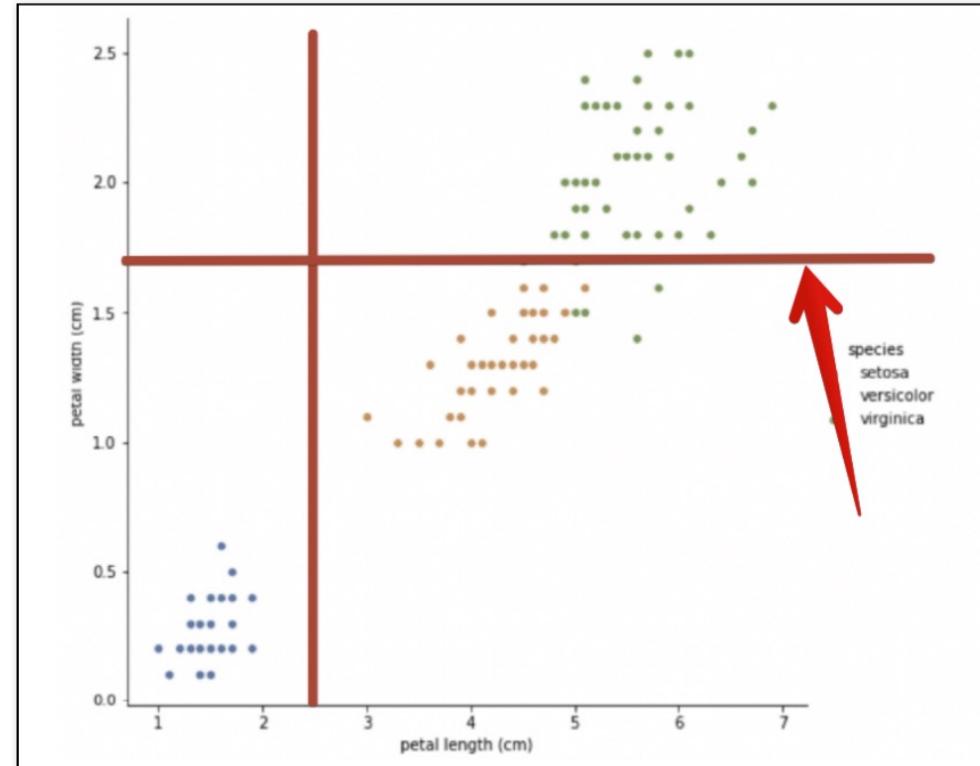
결론?



- 길에서 주운(^^) Iris의 Petal length가 2.5보다 크고,
Petal Width가 1.6보다 크면 Virginica라고 결론을 내려도 될까?
- 누군가 그게 최선이냐고 물으면 뭐라고 대답해야 할까?
- “딱 봐도 그냥 그렇잖아요?”라고 하기엔....ㅠㅠ.

이럴 때 알고리즘이 등장한다

- 현재 상황에서는 이것이 최선이라는 근거
- 혹은 이런 방향, 저런 방향으로 진행했을 때의 각각의 차이점에 대한 정량적 수치 제시

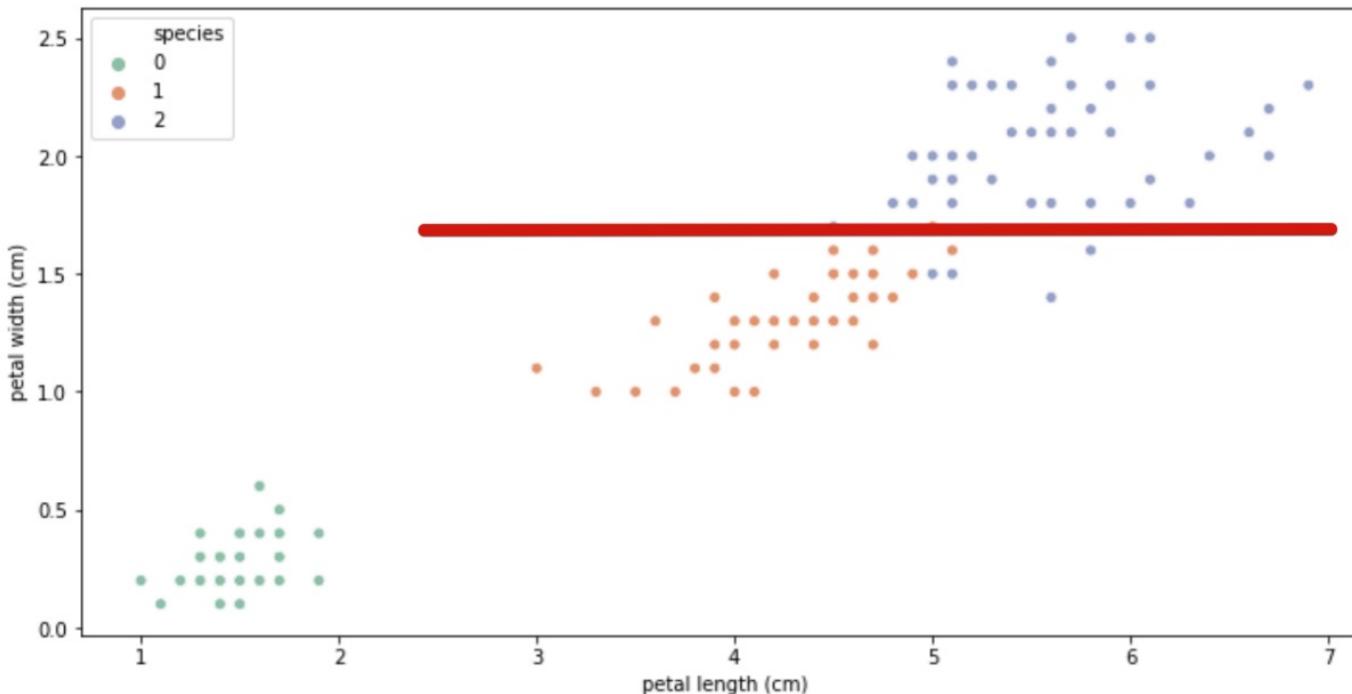


- 저 가로선이 어디에 있는 것이 최선인지 확인해 봅시다

Decision Tree

상황설정

```
plt.figure(figsize=(12, 6))
sns.scatterplot(x="petal length (cm)", y="petal width (cm)",
                 data=iris_pd, hue='species', palette="Set2");
```



- 첫번째 setosa 구분은 너무 잘 되니 두번째 저 선을 어떻게 잘(^^) 찾을 것인가

Decision Tree

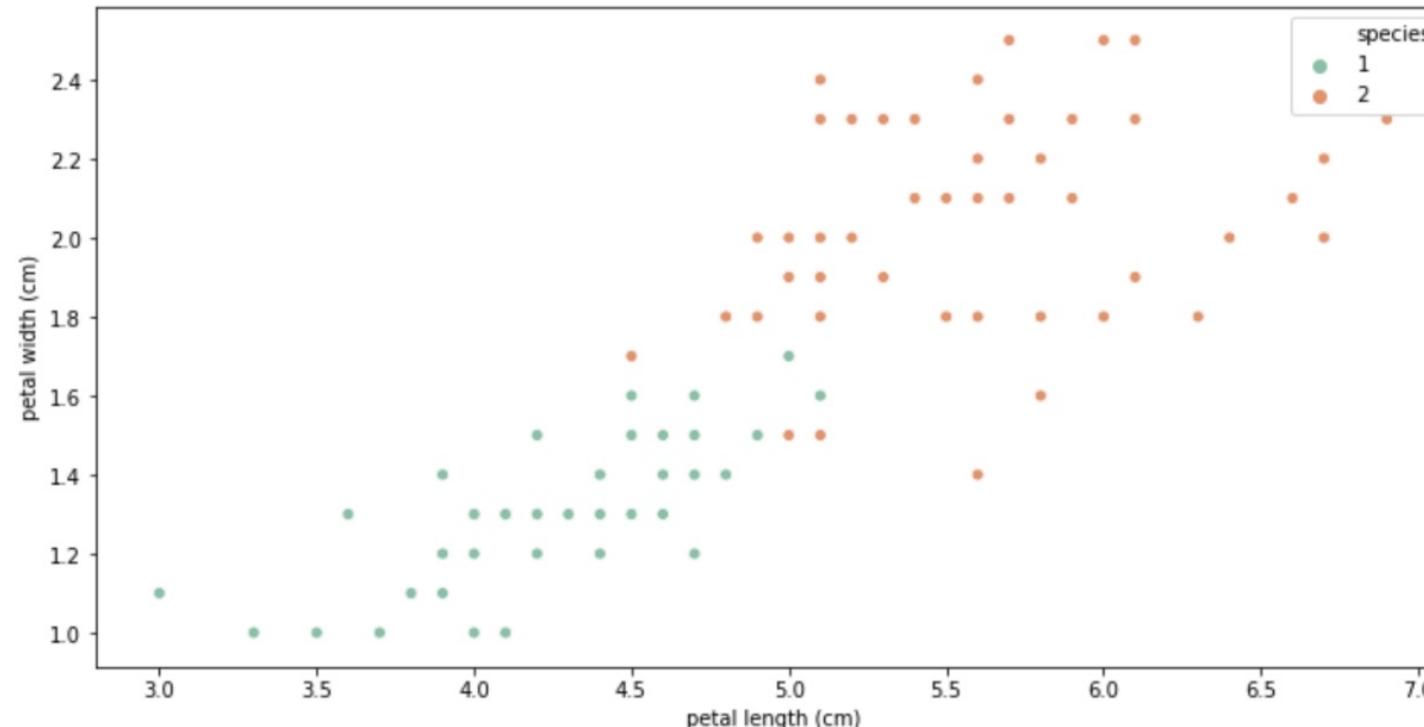
데이터 변경

```
iris_12 = iris_pd[iris_pd['species']!=0]
iris_12.info()

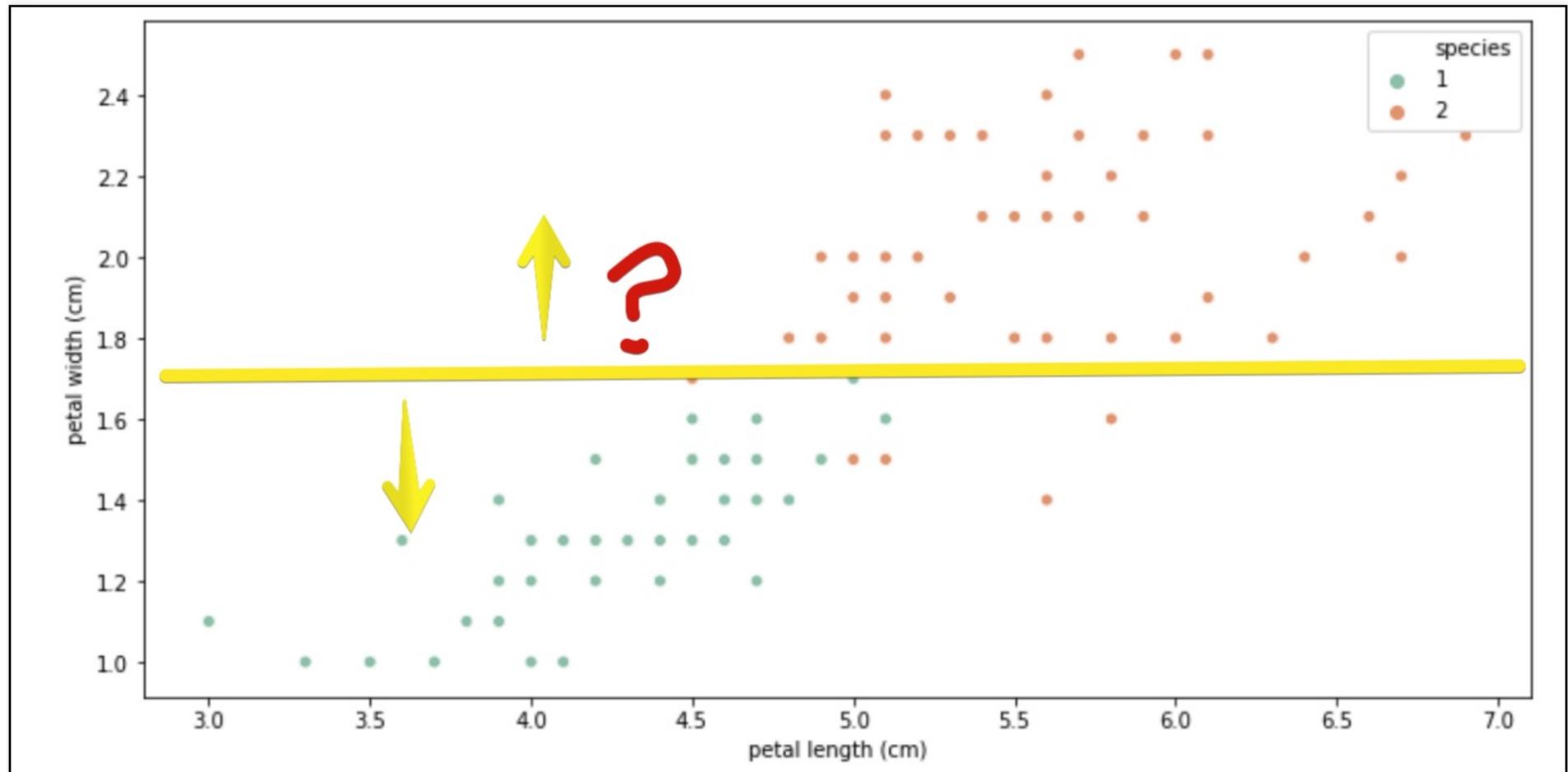
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 50 to 149
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   sepal length (cm)    100 non-null   float64 
 1   sepal width (cm)     100 non-null   float64 
 2   petal length (cm)    100 non-null   float64 
 3   petal width (cm)     100 non-null   float64 
 4   species             100 non-null   int64  
dtypes: float64(4), int64(1)
memory usage: 4.7 KB
```

학습을 위해 두 개의 데이터에 집중해 보자

```
plt.figure(figsize=(12, 6))
sns.scatterplot(x="petal length (cm)", y="petal width (cm)",
                 data=iris_12, hue='species', palette="Set2");
```



다시 저 경계선이 어디에 있으면 최고일까?



Decision Tree의 분할 기준 (Split Criterion)

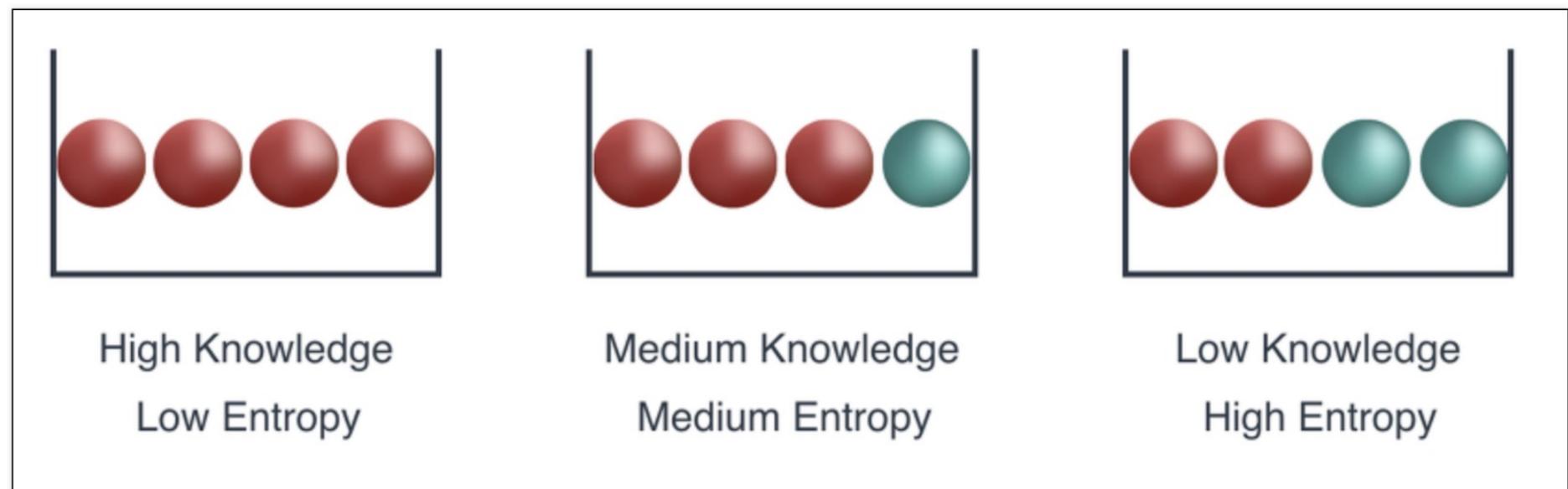
정보 획득 Information Gain

- 보의 가치를 반환하는 데 발생하는 사전의 확률이 작을수록 정보의 가치는 커진다
- 정보 이득이란 어떤 속성을 선택함으로 인해서 데이터를 더 잘 구분하게 되는 것

Decision Tree의 분할
기준 (Split Criterion)

엔트로피 개념

- 열역학의 용어로 물질의 열적 상태를 나타내는 물리 양의 단위 중 하나. 무질서의 정도를 나타냄
- 1948년 엔트로피 개념에서 힌트를 얻어 확률 분포의 무질서도나 불확실성 혹은 정보 부담 정도를 나타내는 정보 엔트로피 개념을 클로드 쇄넌이 고안함



- entropy : 얼마만큼의 정보를 담고 있는가? 또한, 무질서도(disorder)를 의미, 불확실성(uncertainty)을 나타내기도 함

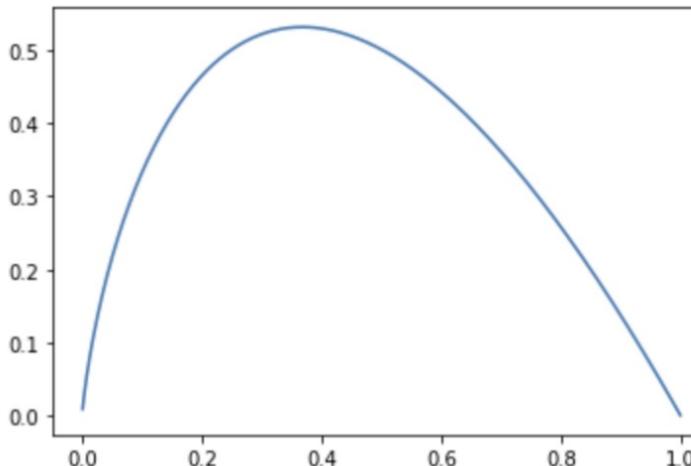
Decision Tree의 분할
기준 (Split Criterion)

$$-p_i \log_2 p_i$$

- p 는 해당 데이터가 해당 클래스에 속할 확률이고 위 식을 그려보면 다음과 같다
- 어떤 확률 분포로 일어나는 사건을 표현하는 데 필요한 정보의 양이며
이 값이 커질수록 확률 분포의 불확실성이 커지며 결과에 대한 예측이 어려워짐

```
import numpy as np

p = np.arange(0.001, 1, 0.001)
plt.plot(p, -p*np.log2(p));
```



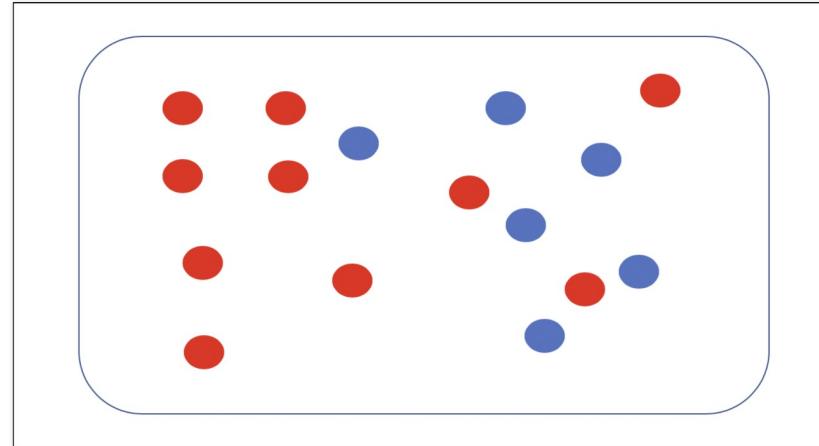
Decision Tree의 분할
기준 (Split Criterion)

- 엔트로피는 이 확률들의 합이다

$$\text{Entropy} = \sum_{k=1}^m -p_i \log_2 p_i$$

Decision Tree의 분할
기준 (Split Criterion)

엔트로피 연습

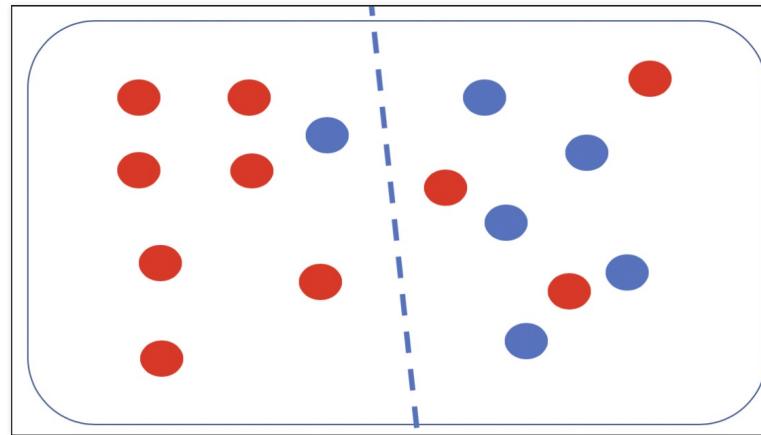


$$\text{Entropy} = \sum_{k=1}^m -p_i \log_2 p_i$$

$$\text{Entropy} = -\frac{10}{16} \log_2 \frac{10}{16} - -\frac{6}{16} \log_2 \frac{6}{16}$$

```
-(10/16)*np.log2(10/16) - 6/16*np.log2(6/16)
```

0.954434002924965

Decision Tree의 분할
기준 (Split Criterion)

$$\text{Entropy} = \sum_{i=1}^d R_i \left\{ \sum_{k=1}^m -p_i \log_2 p_i \right\}$$
$$\text{Entropy} = 0.5 \left\{ -\frac{7}{8} \log_2 \frac{7}{8} - \frac{1}{8} \log_2 \frac{1}{8} \right\} + 0.5 \left\{ -\frac{3}{8} \log_2 \frac{3}{8} - \frac{5}{8} \log_2 \frac{5}{8} \right\}$$

```
| 0.5*(-(7/8)*np.log2(7/8) - 1/8*np.log2(1/8)) + \
| 0.5*(-(3/8)*np.log2(3/8) - 5/8*np.log2(5/8))
```

0.7489992230622807

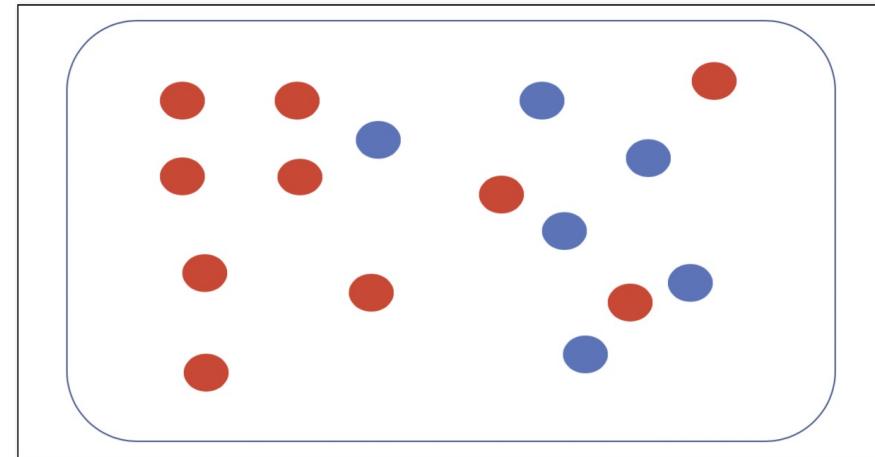
- 엔트로피가 내려갔으므로 분할 하는 것이 좋다.

Decision Tree의 분할
기준 (Split Criterion)

지니 계수

$$Gini = \sum_{k=1}^d R_i \left\{ 1 - \sum_{k=1}^m p_{ik}^2 \right\}$$

- Gini index 혹은 불순도율
- 엔트로피의 계산량이 많아서 비슷한 개념이면서 보다 계산량이 적은 지니계수를 사용하는 경우가 많다.

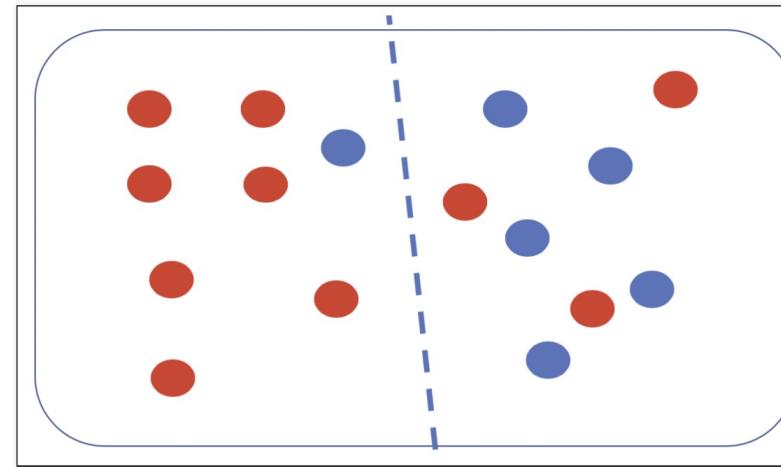
Decision Tree의 분할
기준 (Split Criterion)

$$Gini = 1 - \sum_{k=1}^m p_{ik}^2$$

$$1 - (6/16)^2 - (10/16)^2$$

0.46875

- 분할하지 않았을 때 지니계수는 0.46

Decision Tree의 분할
기준 (Split Criterion)

$$Gini = \sum_{k=1}^d R_i \left\{ 1 - \sum_{k=1}^m p_{ik}^2 \right\}$$

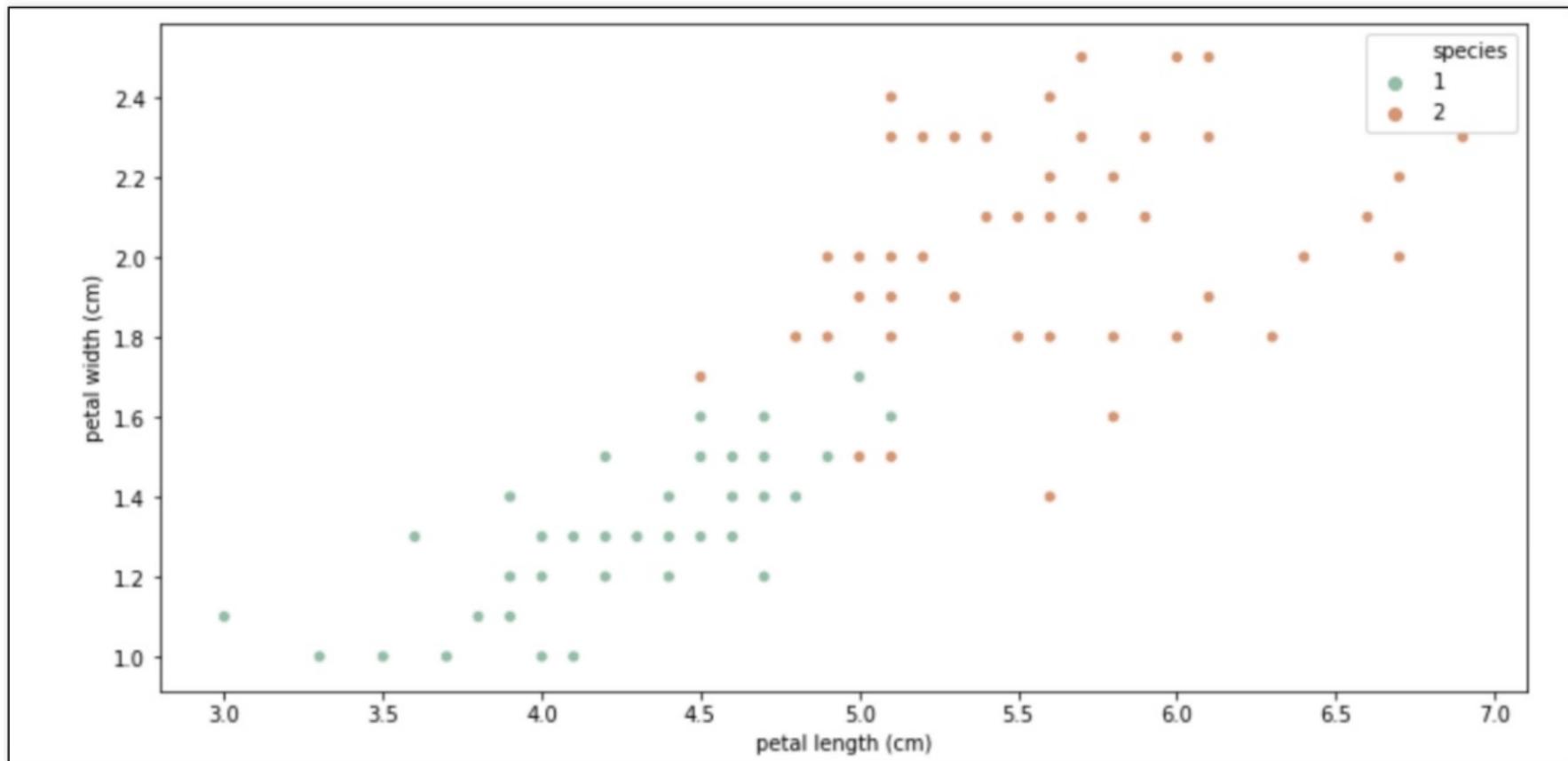
$$0.5 * (1 - (7/8)^{**2} - (1/8)^{**2}) + 0.5 * (1 - (3/8)^{**2} - (5/8)^{**2})$$

0.34375

- 분할 했을 때 지니계수는 0.34
- 분할하는 것으로 결론을 내릴 수 있다.

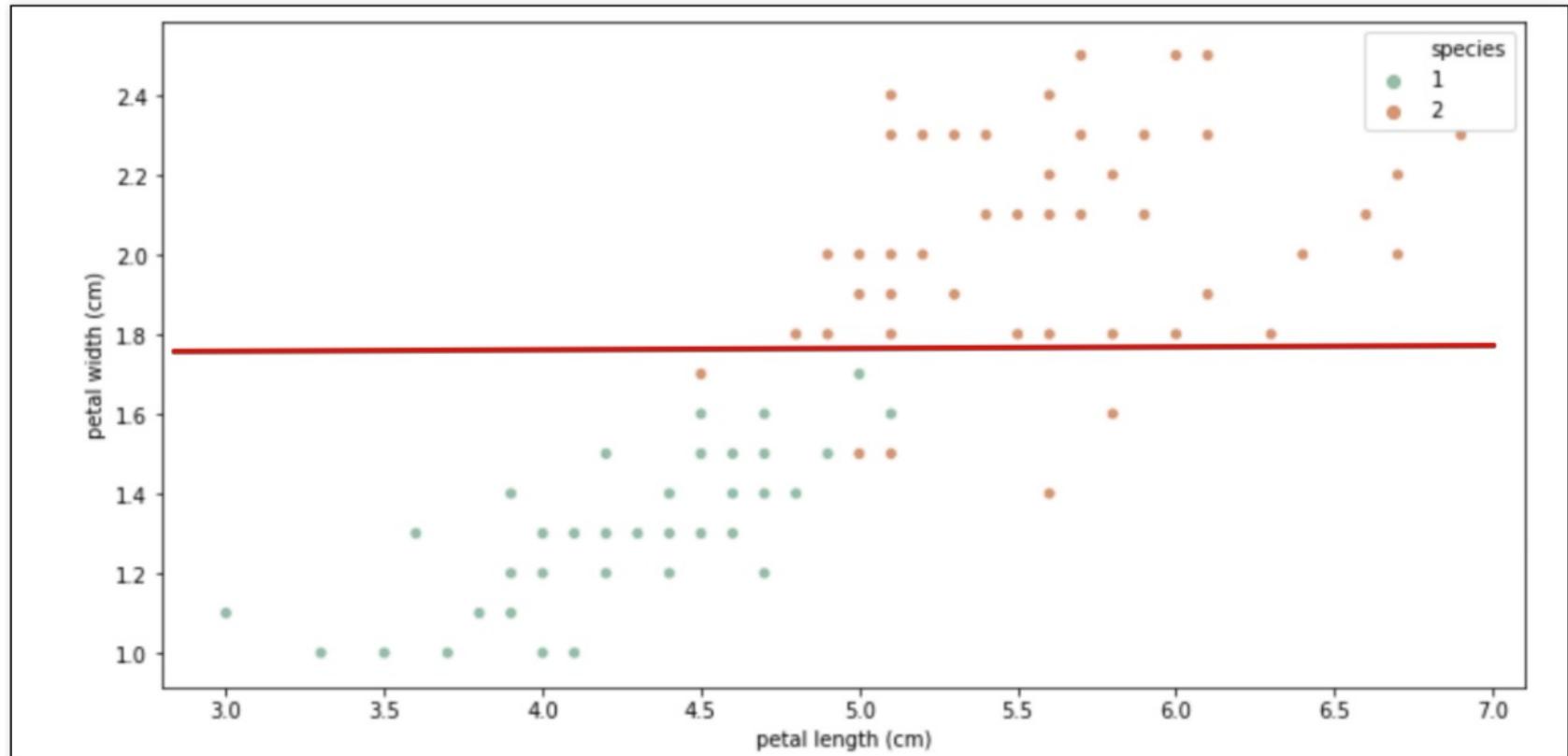
Decision Tree의 분할
기준 (Split Criterion)

그럼 어떻게 가르면 되지?



Decision Tree의 분할
기준 (Split Criterion)

대충 이렇게?



Decision Tree의 분할 기준 (Split Criterion)

Frame Work?

- 근데 이걸 코드로 다 작성한다구요?
- 네... 불과 5~6년? 전만해도 다들 그렇게 했습니다.
- 그 때는 하나의 Lab에서만 사용되는 코드 체계가 있고 이를 자기들끼리 공유하며 사용했었죠.
- 그러다가 최근 SW적 개발 도구의 확산과 공유 등으로 공개적인 Frame Work으로 발전해가기 시작했습니다.

Scikit Learn

Scikit Learn 소개



- 2007년 구글 썸머 코드에서 처음 구현
- 현재 파이썬에서 가장 유명한 기계 학습 오픈 소스 라이브러리

sklearn을 이용한 결정나무의 구현

```
| from sklearn.tree import DecisionTreeClassifier  
  
iris_tree = DecisionTreeClassifier()  
iris_tree.fit(iris.data[:, 2:], iris.target)  
  
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',  
                      max_depth=None, max_features=None, max_leaf_nodes=None,  
                      min_impurity_decrease=0.0, min_impurity_split=None,  
                      min_samples_leaf=1, min_samples_split=2,  
                      min_weight_fraction_leaf=0.0, presort='deprecated',  
                      random_state=None, splitter='best')
```

딸랑 세 줄?



Accuracy 확인

```
from sklearn.metrics import accuracy_score  
  
y_pred_tr = iris_tree.predict(iris.data[:, 2:])  
accuracy_score(iris.target, y_pred_tr)
```

```
0.9933333333333333
```

- 당연히 이게 전부는 아닙니다.
- 이제 우리는 꽤 긴 여정을 시작해야 합니다.
- 단순히 iris 꽃 분류과, 알고리즘 중에서 쉬운 측에 속하는 결정나무이지만, 우리는 좀 먼 여정을 시작해야 합니다.