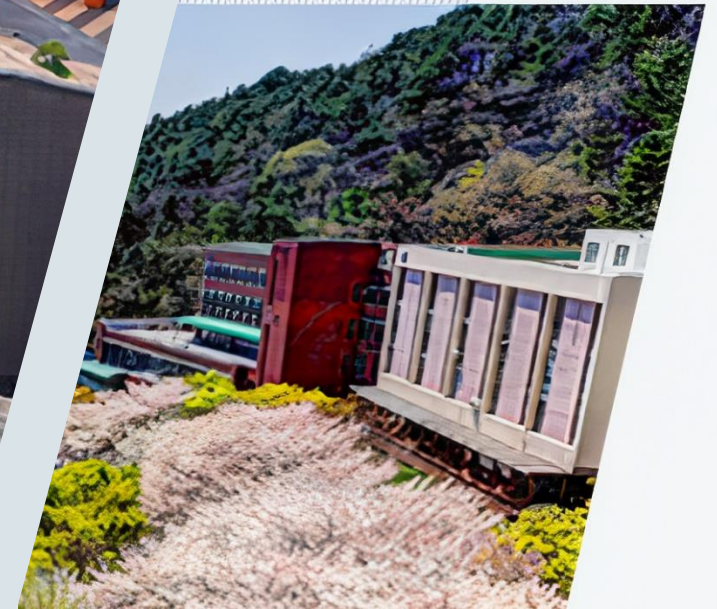


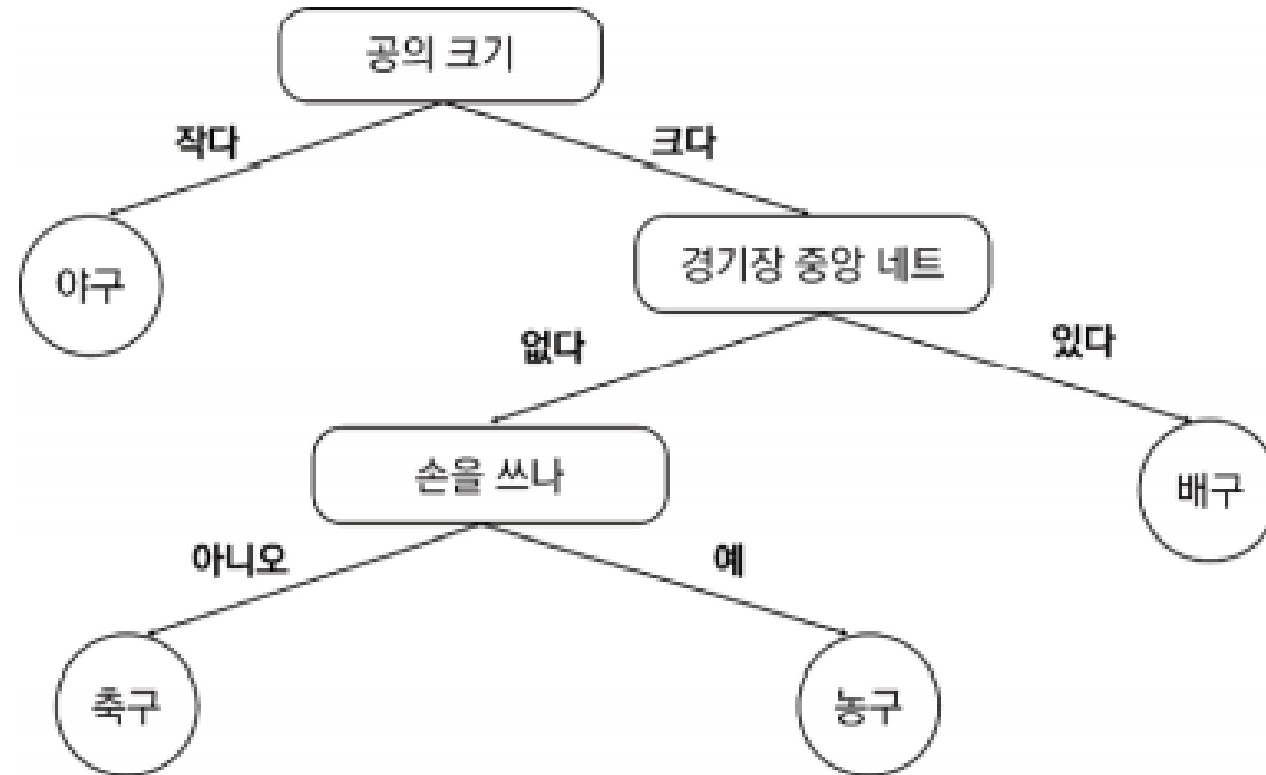
K Nearest Neighbors – 실습

컴퓨터AI공학부
2025년 1학기 머신러닝



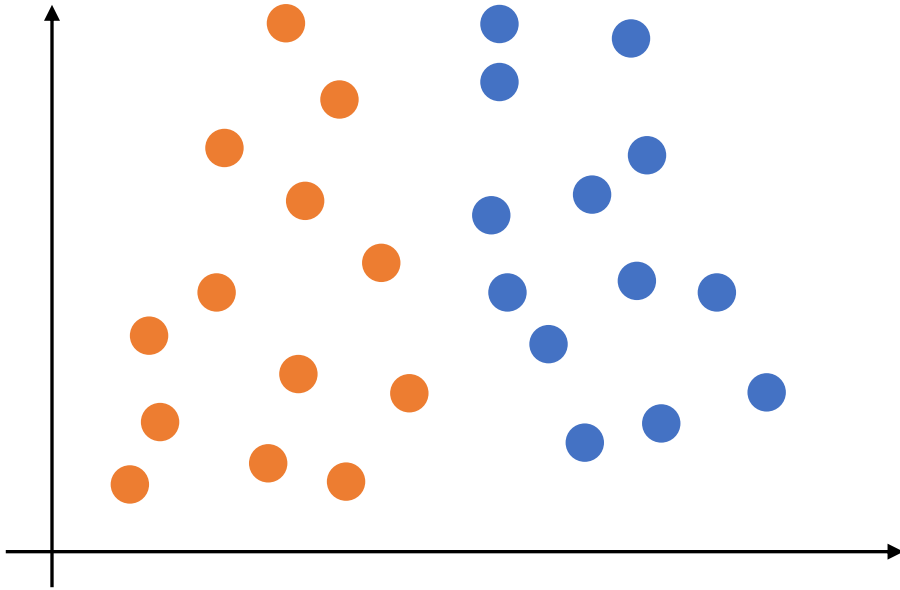
Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Ex) 야구, 배구, 축구, 농구 분류



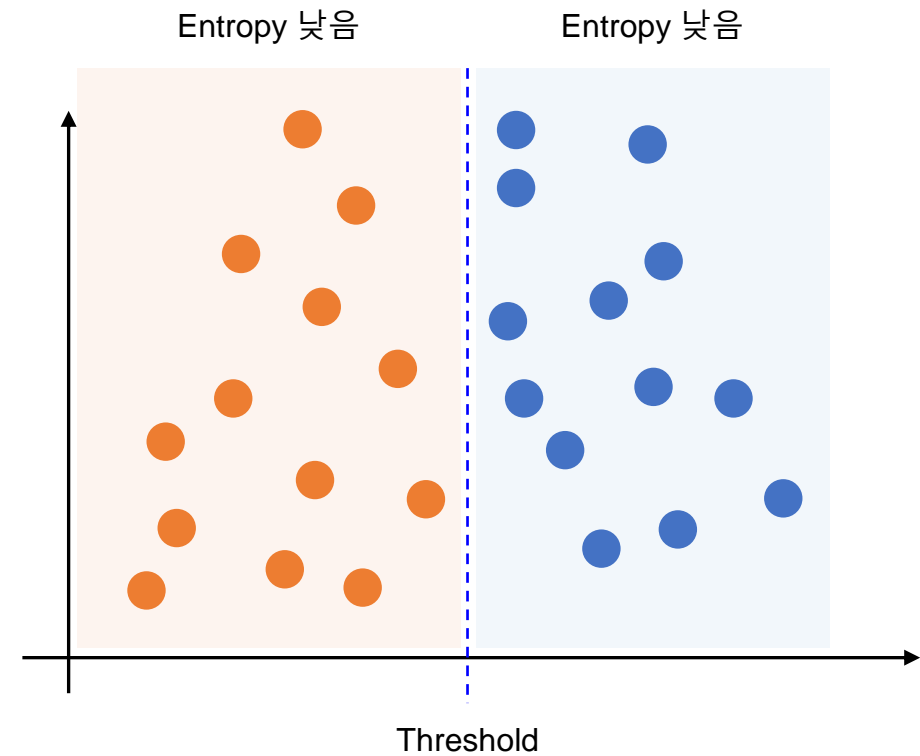
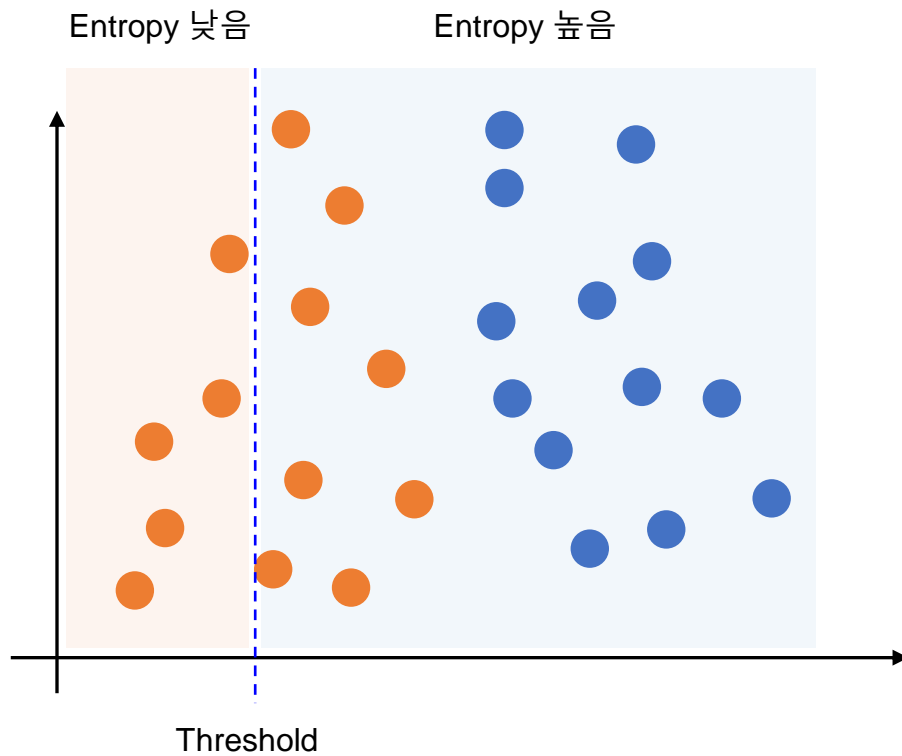
Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Entropy가 감소하는 Threshold (기준점)을 찾아내는 것이 목적



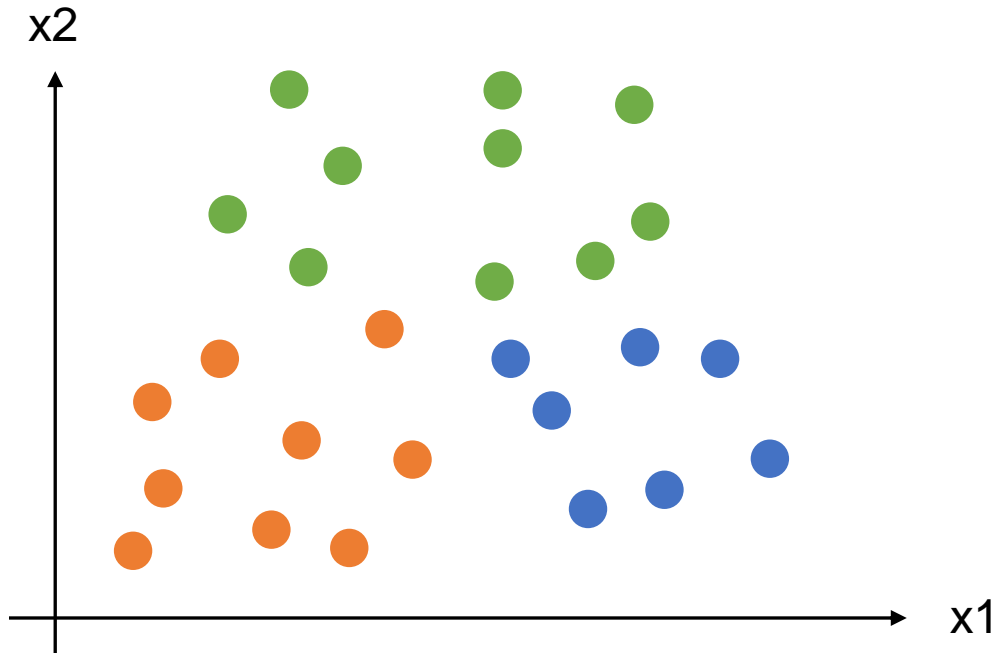
Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Entropy가 감소하는 Threshold (기준점)을 찾아내는 것이 목적



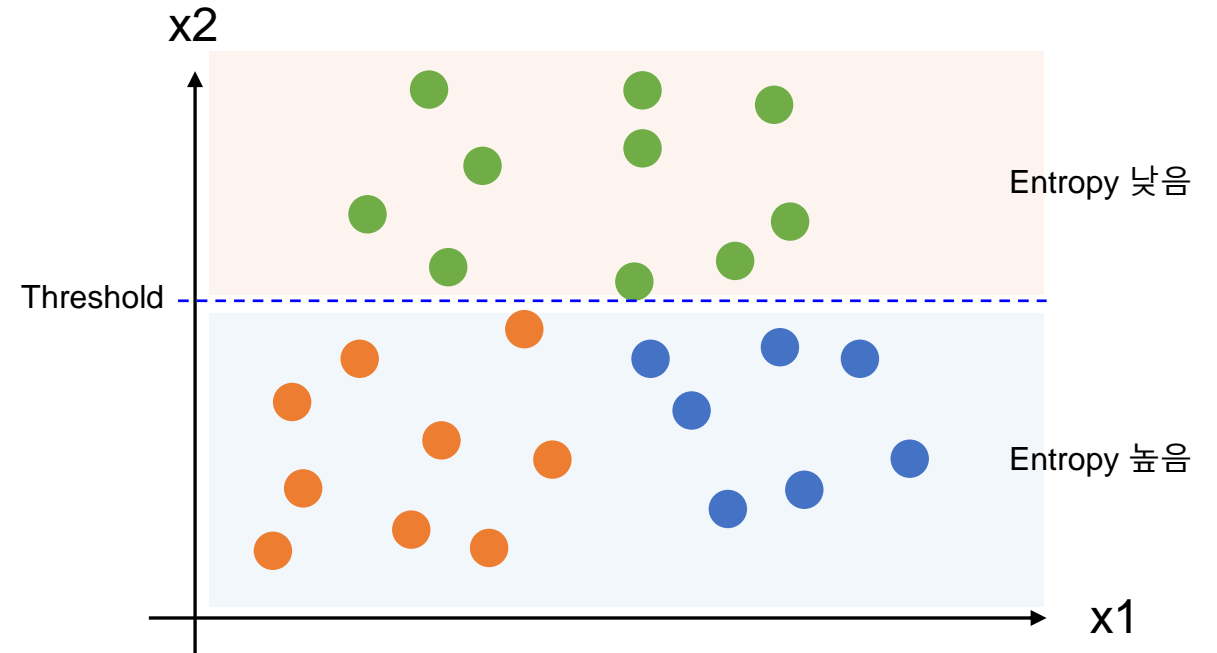
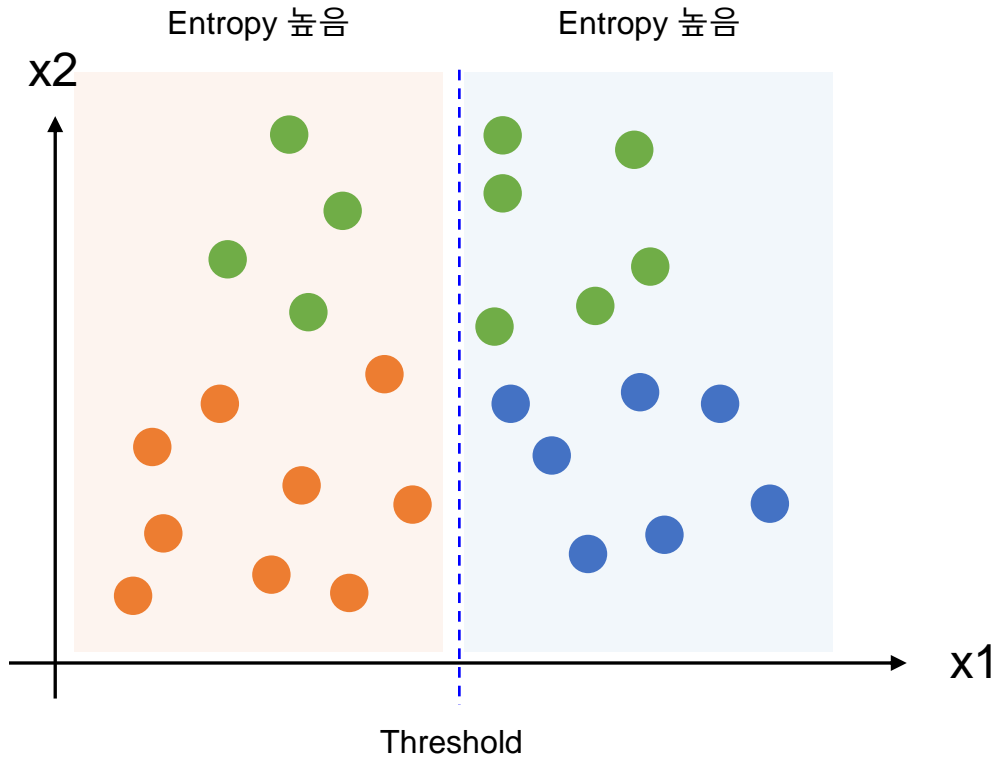
Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Ex) 2개의 특성, 3개의 class를 가지는 데이터를 Decision tree를 이용해 분류



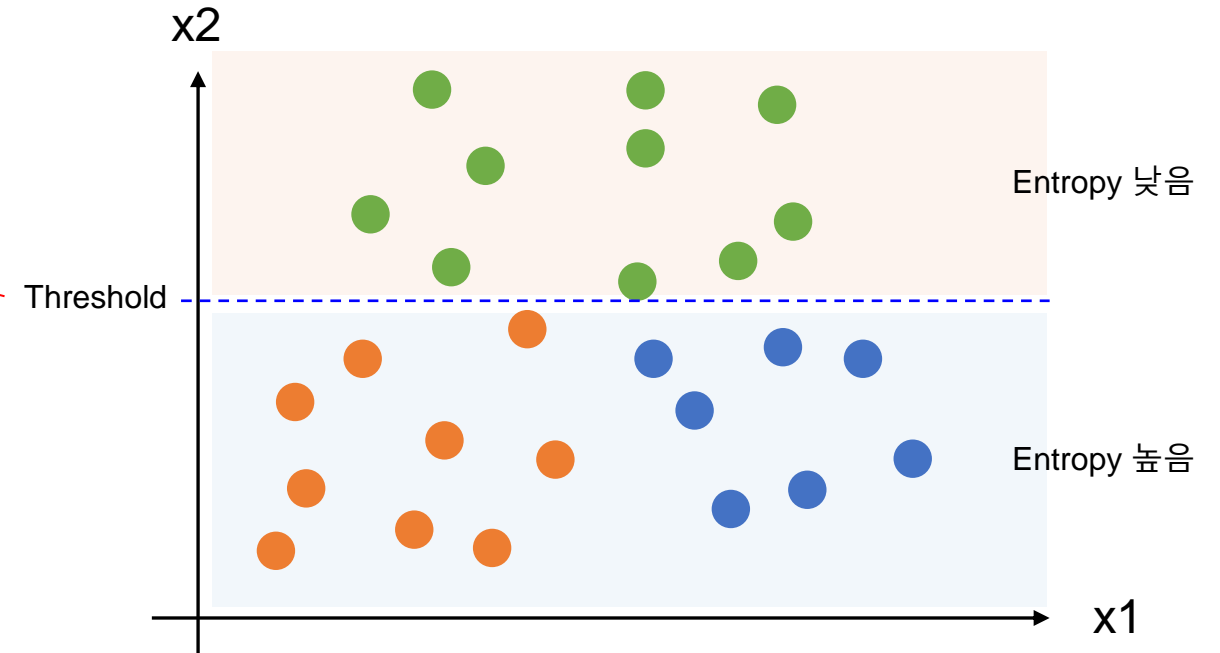
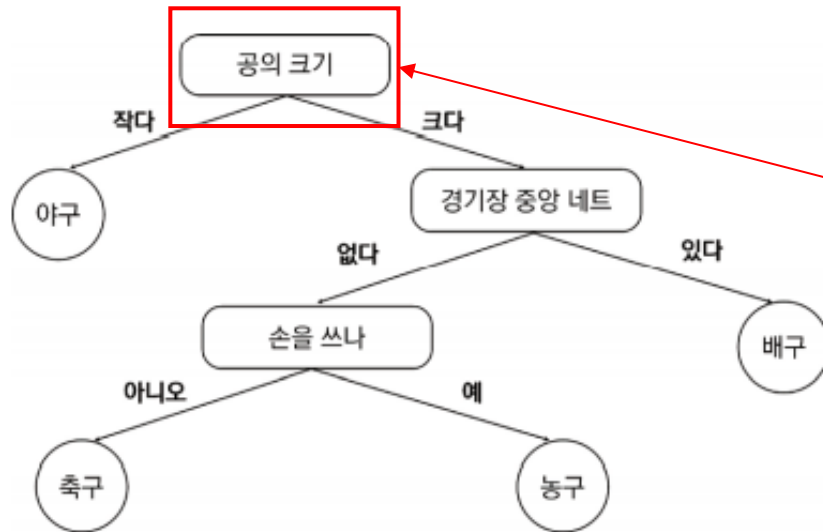
Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Ex) 2개의 특성, 3개의 class를 가지는 데이터를 Decision tree를 이용해 분류



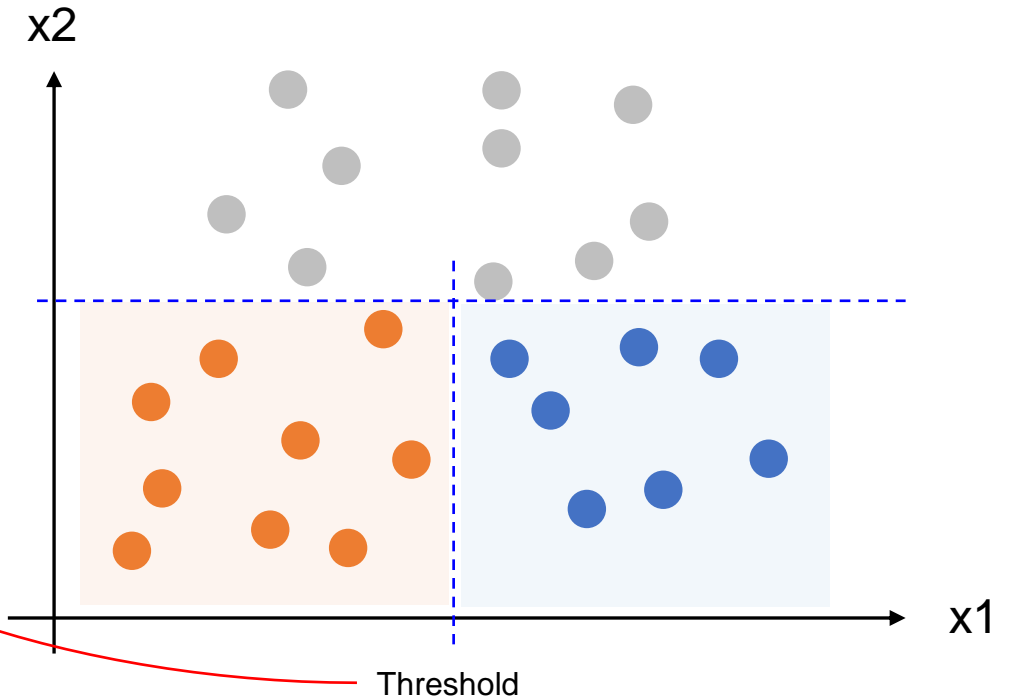
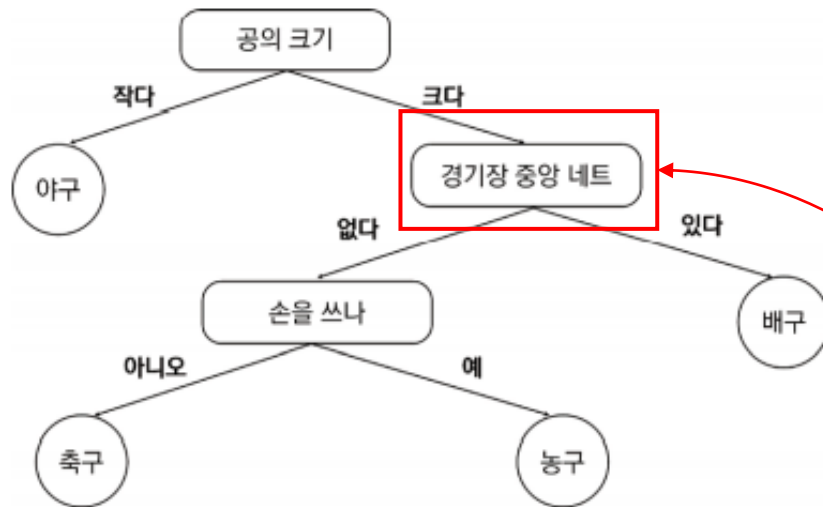
Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Ex) 2개의 특성, 3개의 class를 가지는 데이터를 Decision tree를 이용해 분류



Review – Decision tree

- 목적: 데이터에 있는 규칙을 학습을 통해 찾아내 Tree 기반의 분류 규칙 생성
 - Ex) 2개의 특성, 3개의 class를 가지는 데이터를 Decision tree를 이용해 분류



[실습] Decision Tree (DT)

■ Basecode 다운로드

Dataset 다운로드

```
1 iris = load_iris()
2
3 X = iris.data
4 y = iris.target
5 target_names = iris.target_names
6
7
8 df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
9 df['target'] = iris['target']
10 df['class_name'] = df['target'].apply(lambda idx : iris['target_names'][idx])
11 df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	class_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa
...
145	6.7	3.0	5.2	2.3	2	virginica
146	6.3	2.5	5.0	1.9	2	virginica
147	6.5	3.0	5.2	2.0	2	virginica
148	6.2	3.4	5.4	2.3	2	virginica
149	5.9	3.0	5.1	1.8	2	virginica

150 rows × 6 columns



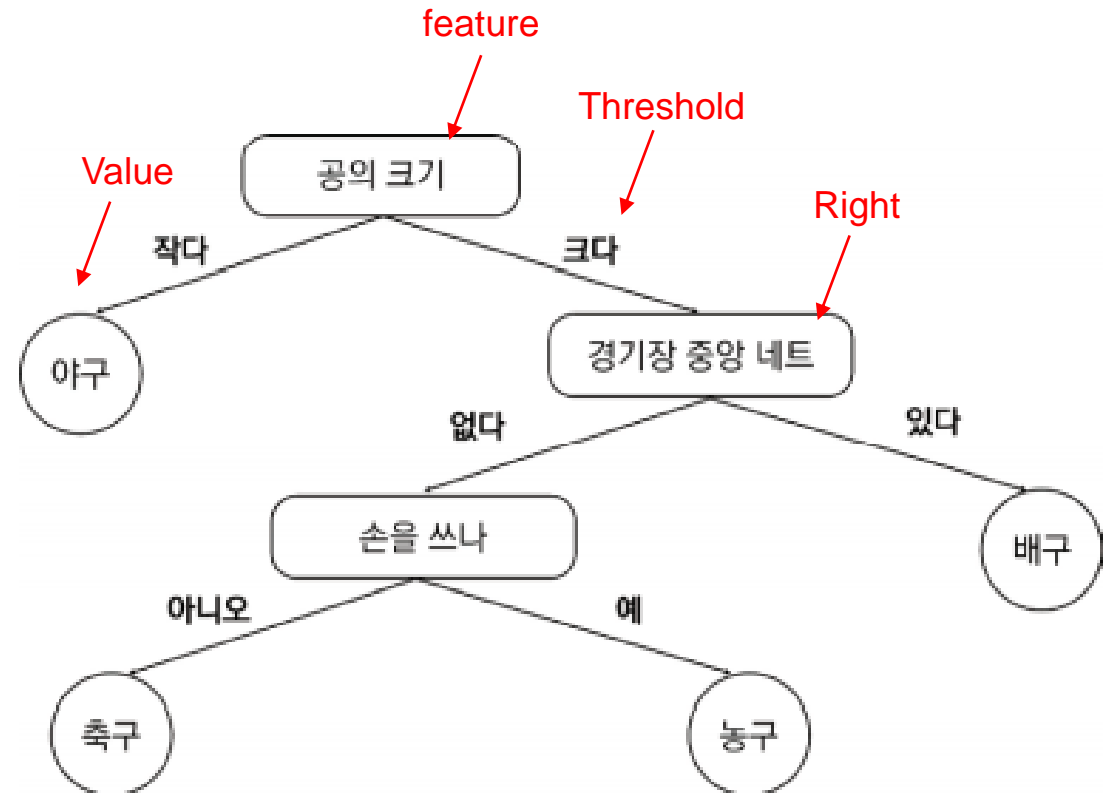
[Petal: 꽃잎, Sepal: 꽃받침]

[실습] Decision Tree (DT)

▪ Basecode 다운로드

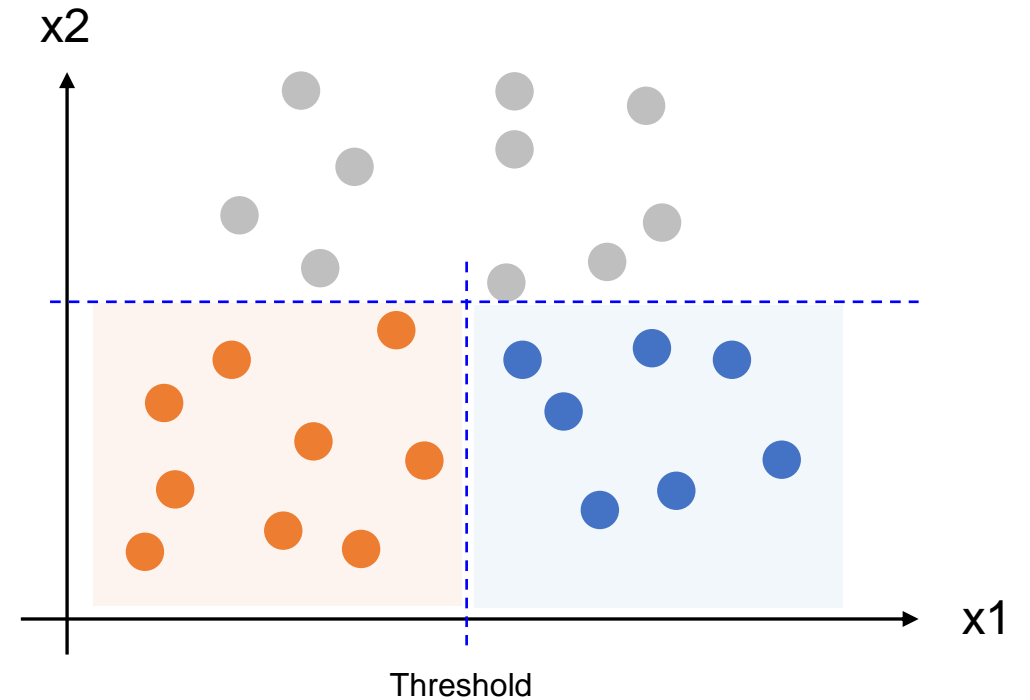
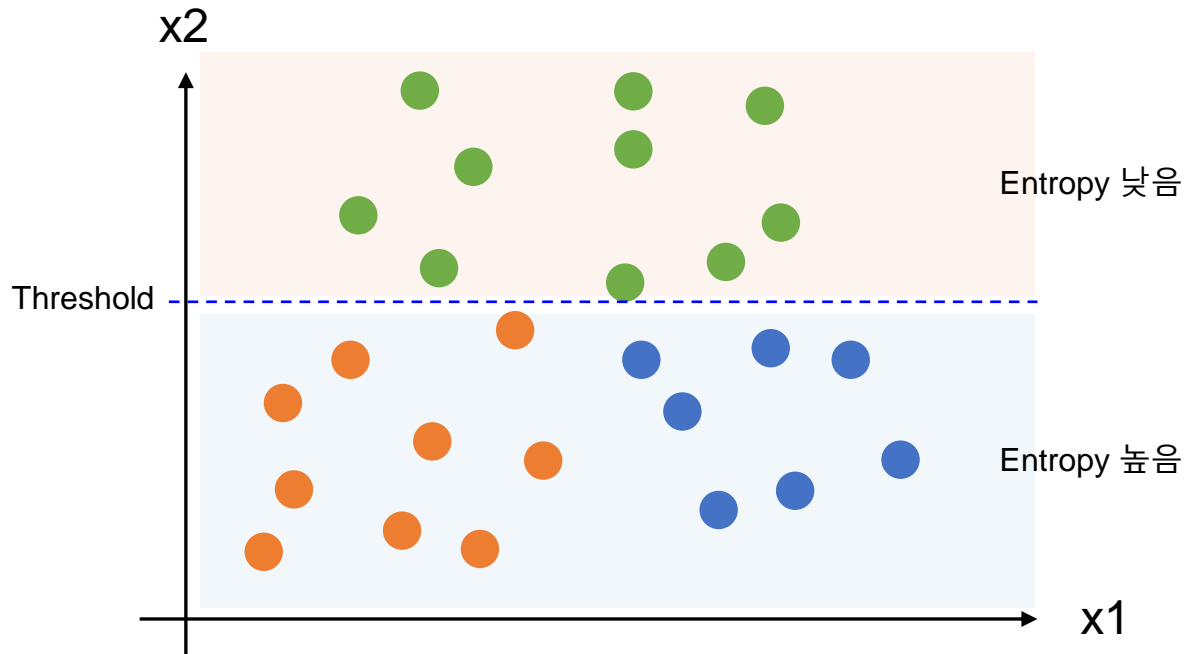
```
class Node:
    def __init__(self, feature=None,
                  threshold=None,
                  left=None,
                  right=None,
                  value=None):

        self.feature = feature
        self.threshold = threshold
        self.left = left
        self.right = right
        self.value = value
```



[실습] Decision Tree (DT)

- Decision tree 코드 작성
 - 재귀 구조를 이용하여 코드 작성



[실습] Decision Tree (DT)

Decision tree 코드 작성

재귀 구조를 이용하여 코드 작성

```
def build_tree(X, y, depth=0):
    # 해당 tree 구역 내의 class의 갯수가 1개의 경우 : 리프노드
    if len(np.unique(y)) == 1:
        return Node(value=y[0])

    # 최적의 특성과 threshold 탐색

    # 예외처리: 어떻게 나뉘도 entropy가 줄어들지 않는 경우 --> 현재 상태 그대로 return
    if feature is None:
        most_common = np.bincount(y).argmax()
        return Node(value=most_common)

    # 탐색한 threshold를 기준으로 데이터 분류
    left_idx
    right_idx

    x_left =
    y_left =

    x_right =
    y_right =

    # 각각의 그룹에 대해 Decision tree 적용
    left =
    right =

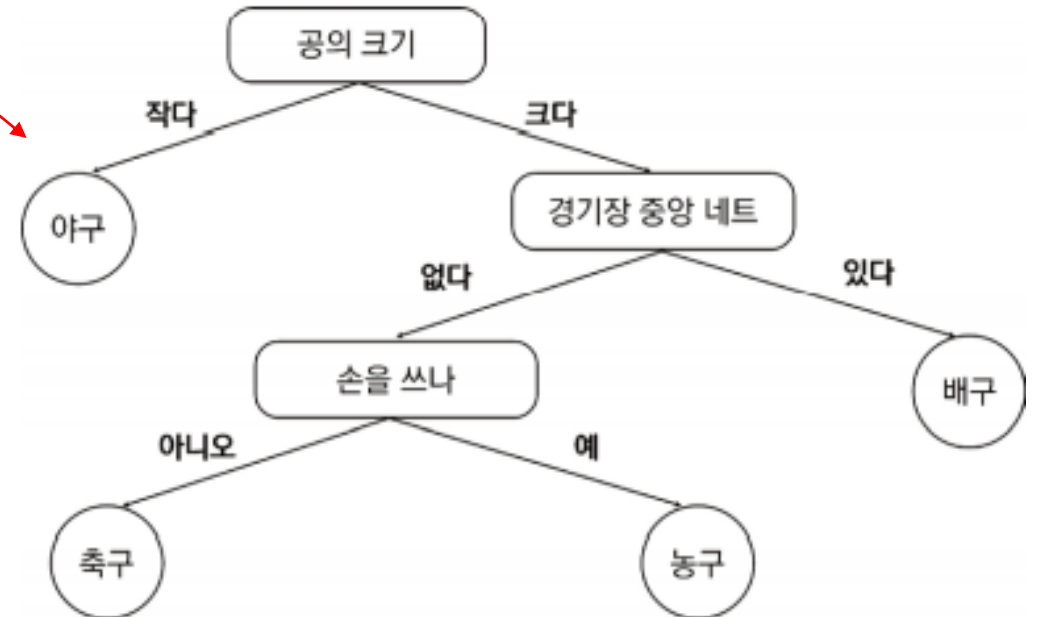
    return Node(feature=feature, threshold=threshold, left=left, right=right)
```

[실습] Decision Tree (DT)

Decision tree 코드 작성

재귀 구조를 이용하여 코드 작성

```
def build_tree(X, y, depth=0):  
    # 해당 tree 구역 내의 class의 갯수가 1개의 경우 : 리프노트  
    if len(np.unique(y)) == 1:  
        return Node(value=y[0])  
  
    # 최적의 특성과 threshold 탐색  
  
    # 예외처리: 어떻게 나눠도 entropy가 줄어들지 않는 경우 --> 현재 상태 그대로 return  
    if feature is None:  
        most_common = np.bincount(y).argmax()  
        return Node(value=most_common)  
  
    # 탐색한 threshold를 기준으로 데이터 분류  
    left_idx  
    right_idx  
  
    x_left =  
    y_left =  
  
    x_right =  
    y_right =  
  
    # 각각의 그룹에 대해 Decision tree 적용  
    left =  
    right =  
  
    return Node(feature=feature, threshold=threshold, left=left, right=right)
```

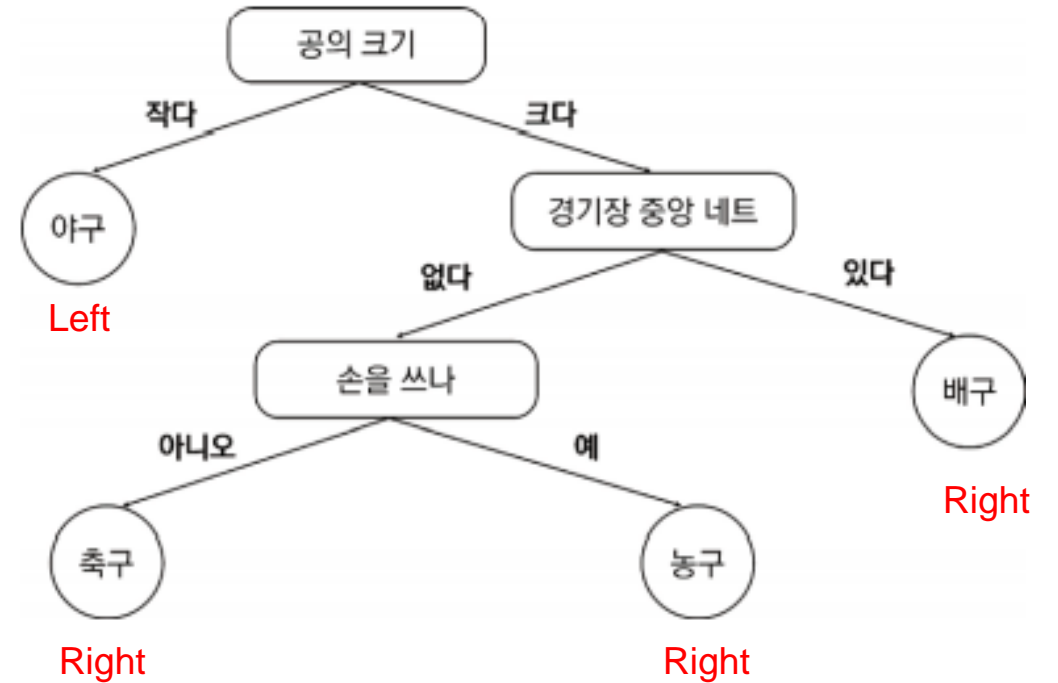


[실습] Decision Tree (DT)

Decision tree 코드 작성

- 재귀 구조를 이용하여 코드 작성

```
def build_tree(X, y, depth=0):  
    # 해당 tree 구역 내의 class의 갯수가 1개의 경우 : 리프노트  
    if len(np.unique(y)) == 1:  
        return Node(value=y[0])  
  
    # 최적의 특성과 threshold 탐색  
  
    # 예외처리: 어떻게 나눠도 entropy가 줄어들지 않는 경우 --> 현재 상태 그대로 return  
    if feature is None:  
        most_common = np.bincount(y).argmax()  
        return Node(value=most_common)  
  
    # 탐색한 threshold를 기준으로 데이터 분류  
    left_idx  
    right_idx  
  
    x_left =  
    y_left =  
  
    x_right =  
    y_right =  
  
    # 각각의 그룹에 대해 Decision tree 적용  
    left =  
    right =  
  
    return Node(feature=feature, threshold=threshold, left=left, right=right)
```

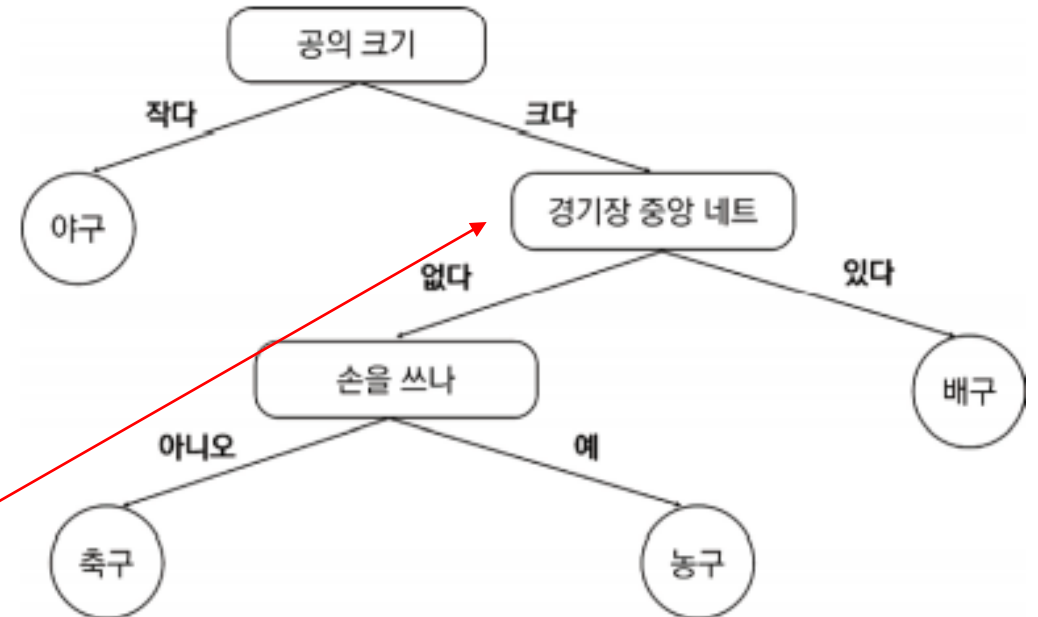


[실습] Decision Tree (DT)

Decision tree 코드 작성

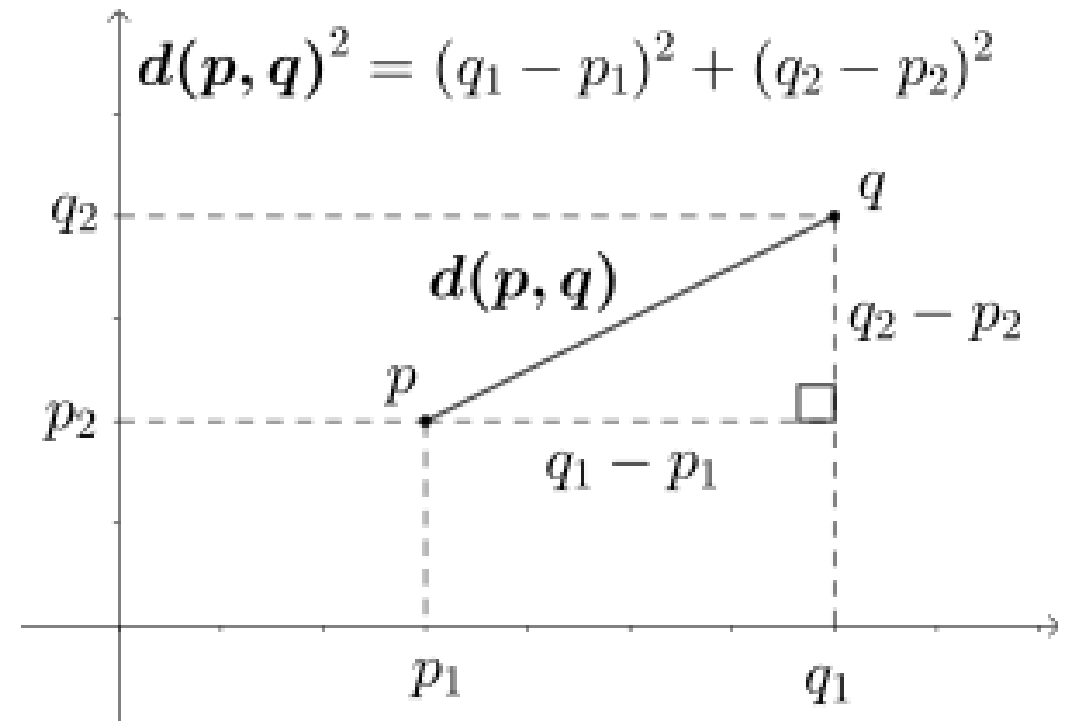
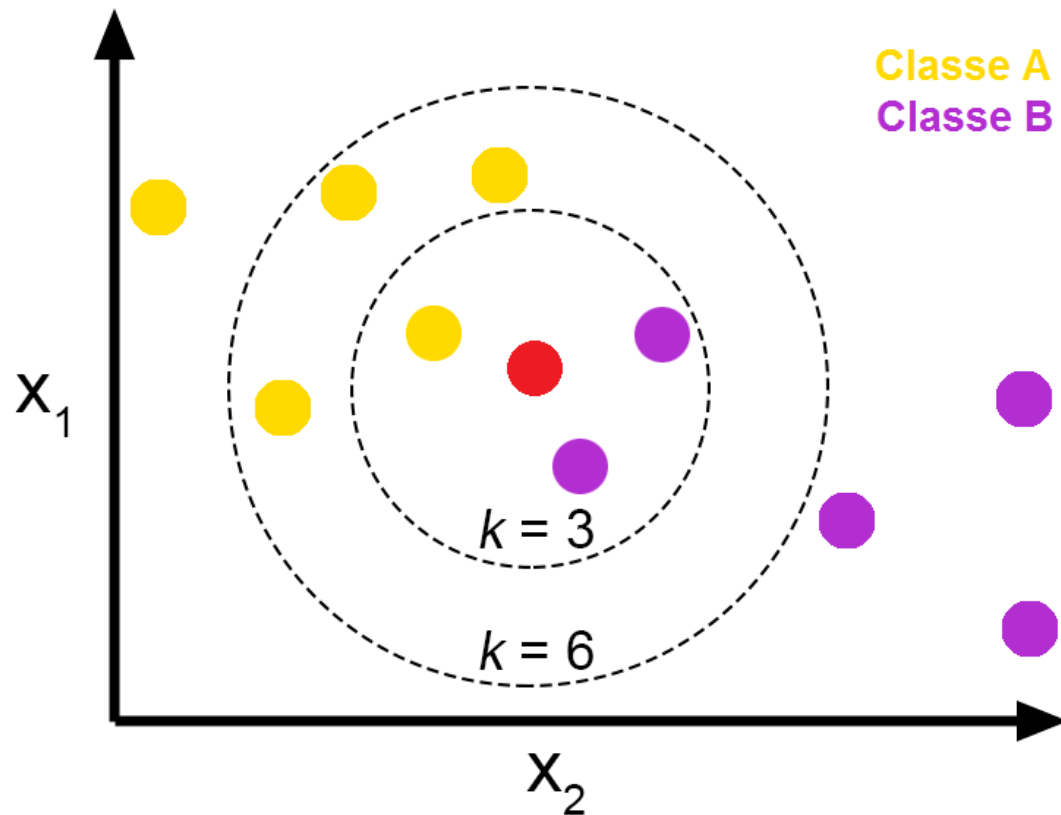
- 재귀 구조를 이용하여 코드 작성

```
def build_tree(X, y, depth=0):  
    # 해당 tree 구역 내의 class의 갯수가 1개의 경우 : 리프노트  
    if len(np.unique(y)) == 1:  
        return Node(value=y[0])  
  
    # 최적의 특성과 threshold 탐색  
  
    # 예외처리: 어떻게 나눠도 entropy가 줄어들지 않는 경우 --> 현재 상태 그대로 return  
    if feature is None:  
        most_common = np.bincount(y).argmax()  
        return Node(value=most_common)  
  
    # 탐색한 threshold를 기준으로 데이터 분류  
    left_idx  
    right_idx  
  
    x_left =  
    y_left =  
  
    x_right =  
    y_right =  
  
    # 각각의 그룹에 대해 Decision tree 적용  
    left =  
    right =  
  
    return Node(feature=feature, threshold=threshold, left=left, right=right)
```



Review – K Nearest Neighbors

- 목적: 새로운 샘플에서 가장 인접한 k개 샘플의 class에 따라 현재 class 분류

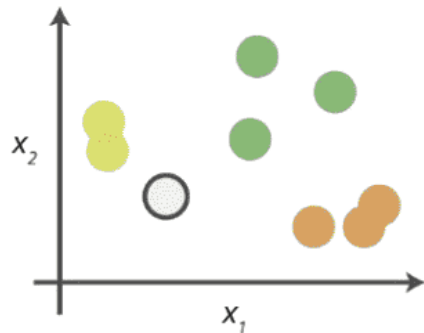


Euclidean distance (L2 distance)

Review – K Nearest Neighbors

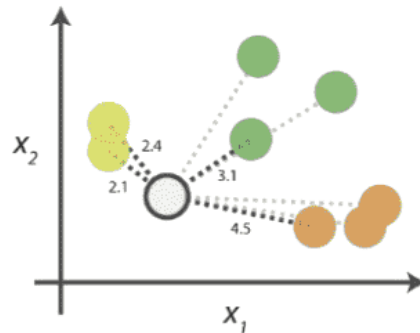
- 목적: 새로운 샘플에서 가장 인접한 k개 샘플의 class에 따라 현재 class 분류

0. Look at the data







Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances









Start by calculating the distances between the grey point and all other points.

2. Find neighbours

Point Distance			
	2.1	→	1st NN
	2.4	→	2nd NN
	3.1	→	3rd NN
	4.5	→	4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels

Class	# of votes	
	2	➔ Class  wins the vote! Point  is therefore predicted to be of class  .
	1	
	1	

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

[실습] K Nearest Neighbors (KNN)

■ Basecode 다운로드

▼ K Nearest Neighbors (KNN)

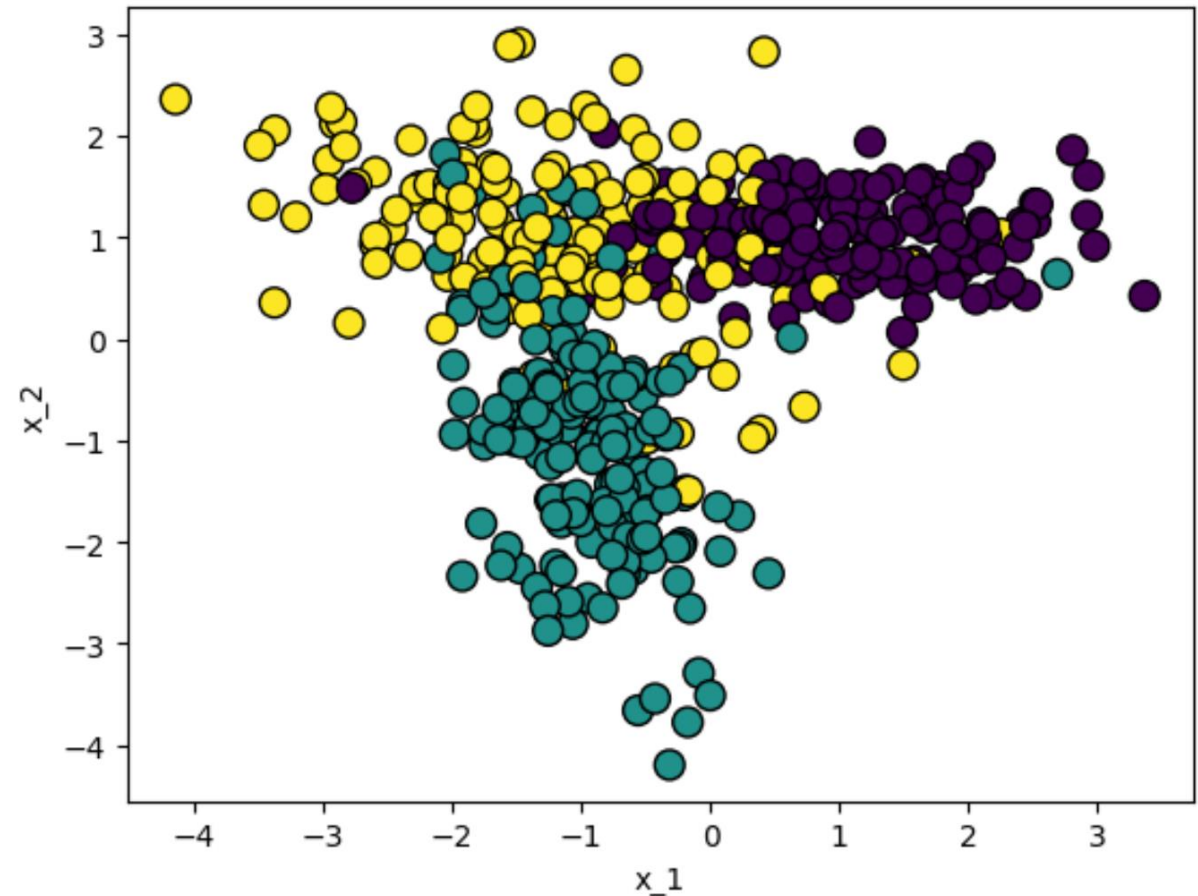
```
[14] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

▼ Dataset

```
[54] X, y = make_classification(n_samples=500,
                               n_features=2,
                               n_classes=3,
                               n_clusters_per_class=1,
                               n_informative=2,
                               n_redundant=0,
                               random_state=40)

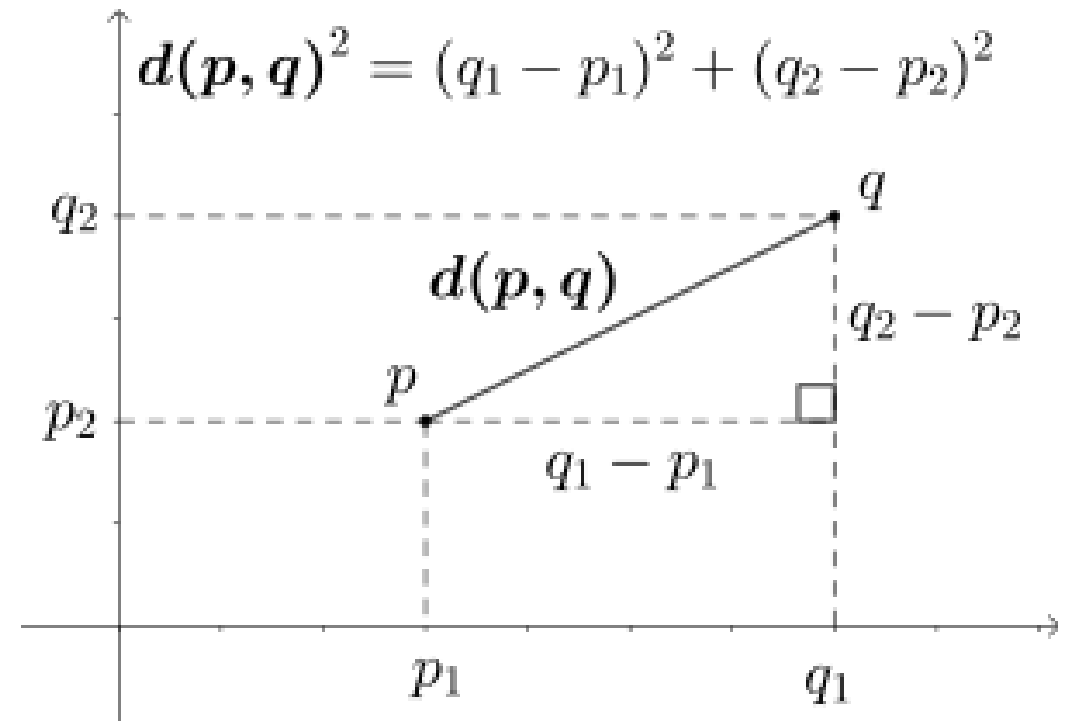
plt.scatter(X[:, 0], X[:, 1], marker='o', c=y, s=100, edgecolor="k", linewidth=1)
plt.xlabel("x_1")
plt.ylabel("x_2")
plt.show()
```



[실습] K Nearest Neighbors (KNN)

- Euclidian Distance 함수, KNN 모델 작성

```
def L2_distance(x1, x2):  
    return np.sqrt(np.sum((x1 - x2) ** 2))  
  
class KNN:  
    def __init__(self, k=3):  
        # initialization  
  
    def fit(self, X, y):  
        # Storage training datas  
  
    def predict(self, X):  
        # Prediction
```



Euclidean distance (L2 distance)

[실습] K Nearest Neighbors (KNN)

- 예측 및 성능 평가

▽ Prediction

```
[56] model = KNN()  
      model.fit(X_train, y_train)  
      y_pred = model.predict(X_test)  
  
      accuracy = np.sum(y_pred == y_test) / len(y_test)  
      print(accuracy)
```

0.81

[실습] K Nearest Neighbors (KNN)

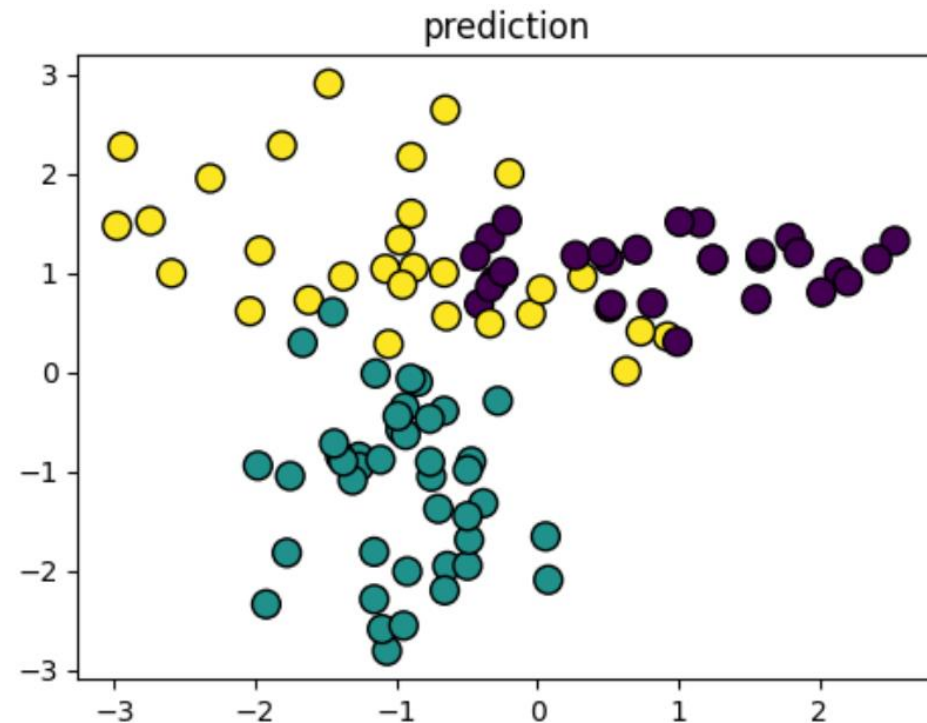
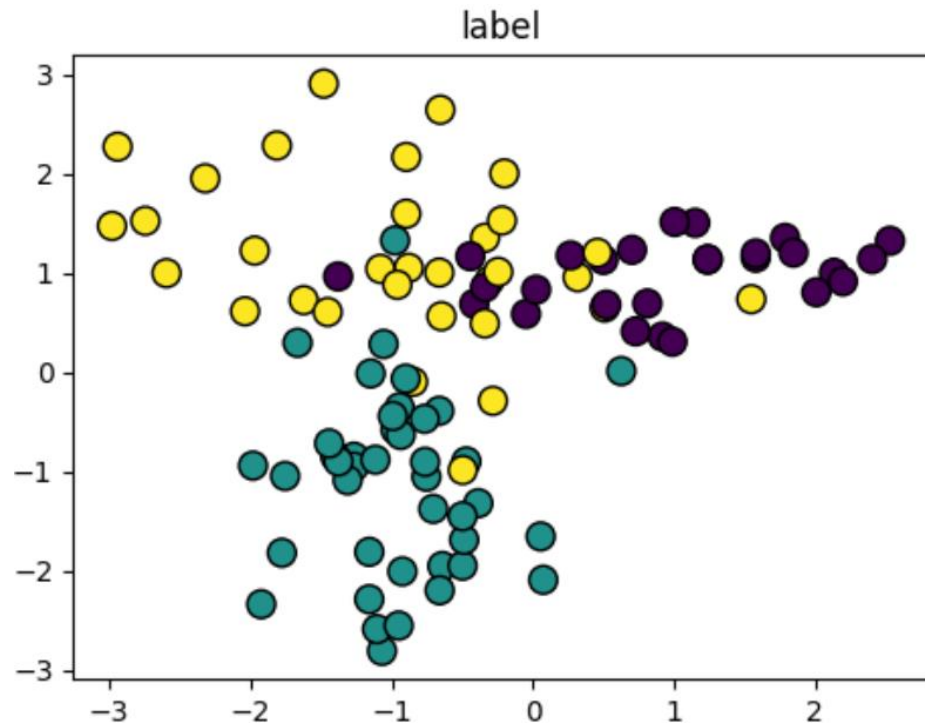
■ 예측 결과 시각화

- Label: 정답 데이터
- Prediction: 예측 값

```
[20] plt.figure(figsize=(12,6))

plt.subplot(1, 2, 1)
plt.title("label")
plt.scatter(X_test[:, 0], X_test[:, 1], marker='o', c=y_test, s=100, edgecolor="k", linewidth=1)

plt.subplot(1, 2, 2)
plt.title("prediction")
plt.scatter(X_test[:, 0], X_test[:, 1], marker='o', c=y_pred, s=100, edgecolor="k", linewidth=1)
plt.show()
```



Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)

Image Signal Processing Laboratory (www.donga-ispl.kr)

Dept. of Computer Engineering

Dong-A University, Busan, Rep. of Korea