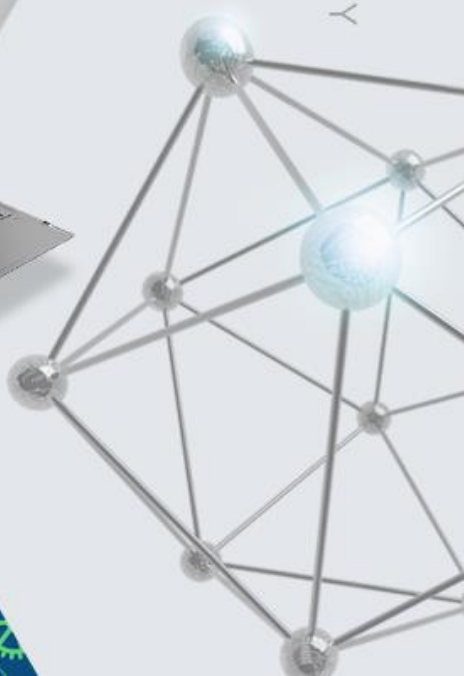
한국기술교육대학교
온라인평생교육원

The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

TECHNOLOGY

딥러닝 입문

인공지능 기초



The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

인공지능 기초

학/습/목/표

1. 기계학습의 개념에 대해 이해하고, 지도학습과 비지도학습을 구분할 수 있다.
2. 지도학습의 개념에 대해 이해하고, 직접 코딩할 수 있다.
3. 비지도학습의 개념에 대해 이해하고, 직접 코딩할 수 있다.

학/습/내/용

1. 기계학습
2. 지도학습
3. 비지도학습

1. 기계학습

1) 기계학습의 개념

(1) 기계학습

- 컴퓨터에게 명시적으로 프로그램을 하지 않고도 컴퓨터가 학습을 알 수 있도록 능력을 갖는 것
- 스스로 학습하는 인공지능 기술, 사람이 아닌 컴퓨터가 직접 프로그램을 만드는 것

1. 기계학습

2) 기계학습의 발전 과정

(1) 초창기 인공지능 : 지식기반

- 지식기반 : 사람이 잘 아는 것을 컴퓨터에게도 기대
- 사람은 변화를 쉽게 인지하지만 컴퓨터는 그렇지 못함
  (패턴을 찾기 어려움)

(2) 기계학습의 등장

- 지식기반 방식의 한계를 깨달으며 기계학습이 등장
- 지식기반 접근방식 >> 데이터 중심 접근방식 (기계학습의 첫 번째 시작)

3) 기계학습의 개념 – 기존 프로그래밍 방식

(1) 사람이 직접 입력 데이터와 프로그램을 작성하여 원하는 결과 받기

10만원으로 사과 10개를 사고, 거스름돈이 45,000원일 때
사과 1개의 가격은?

4) 기계학습의 개념 – 기계학습 방식

(1) 컴퓨터에게 입력과 결과를 함께 알려주고 스스로 프로그램을 만들 수 있게 하기

10만원으로 사과 10개를 사고, 거스름돈이 45,000원

5만원으로 사과 2개를 사고, 거스름돈이 39,000원

2만원으로 사과 3개를 사고, 거스름돈이 3,500원

- 기계학습은 조건을 사용하여 특정 경우마다 답을 찾아주는 것이 아닌 조건을 컴퓨터 학습 모델의 가중치로 바꿔서 학습이 되게 하는 방식임
- 조건을 사용하여 특정 경우마다 답을 찾아주는 방식의 경우
  (사람이 알고 있는 것을 그대로 컴퓨터에게 프로그래밍)
  - 새로운 입력이 들어왔을 때 컴퓨터가 문제를 해결하지 못하는 단점 존재

2. 지도학습

1) 지도학습의 정의

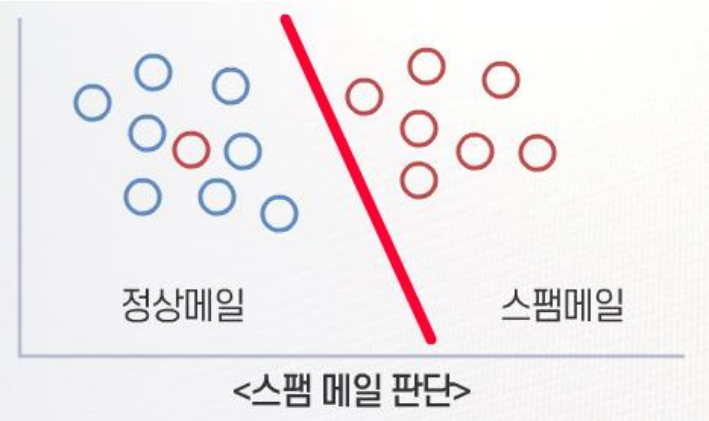
(1) 지도학습(Supervised learning)이란?

- 입력데이터와 결과데이터를 함께 모두 알려주고 컴퓨터에게 답을 찾을 수 있는 프로그램을 스스로 찾을 수 있도록 학습시키는 것
- 목적 값(Target Value, 정답)이 존재함
- 알려준 정답을 토대로 나름대로의 특징을 찾기 위해 학습함
  - 판단을 하기 위해 수많은 데이터 학습(분류방법, 예측방법)

2) 지도학습의 분류방법

(1) 지도학습의 분류방법이란?

- 이전까지의 컴퓨터가 학습한 데이터를 기초로 새로운 데이터가 들어왔을 때 새로운 데이터를 분류하는 방법
  - 주로 입력과 결과데이터의 연속성이 없는 경우에 많이 사용함



<스팸 메일 판단>

정상/스팸 메일 데이터를 기반으로 학습한 뒤
새로운 메일을 알려줬을 때 스팸인지 아닌지 분류

2. 지도학습

3) 지도학습의 예측방법

(1) 지도학습의 예측방법이란?

- 이전까지의 컴퓨터가 학습한 데이터를 기초로 새로운 데이터를 예측하는 방법

  - 주로 입력과 결과데이터의 연속성이 있는 경우에 많이 사용함

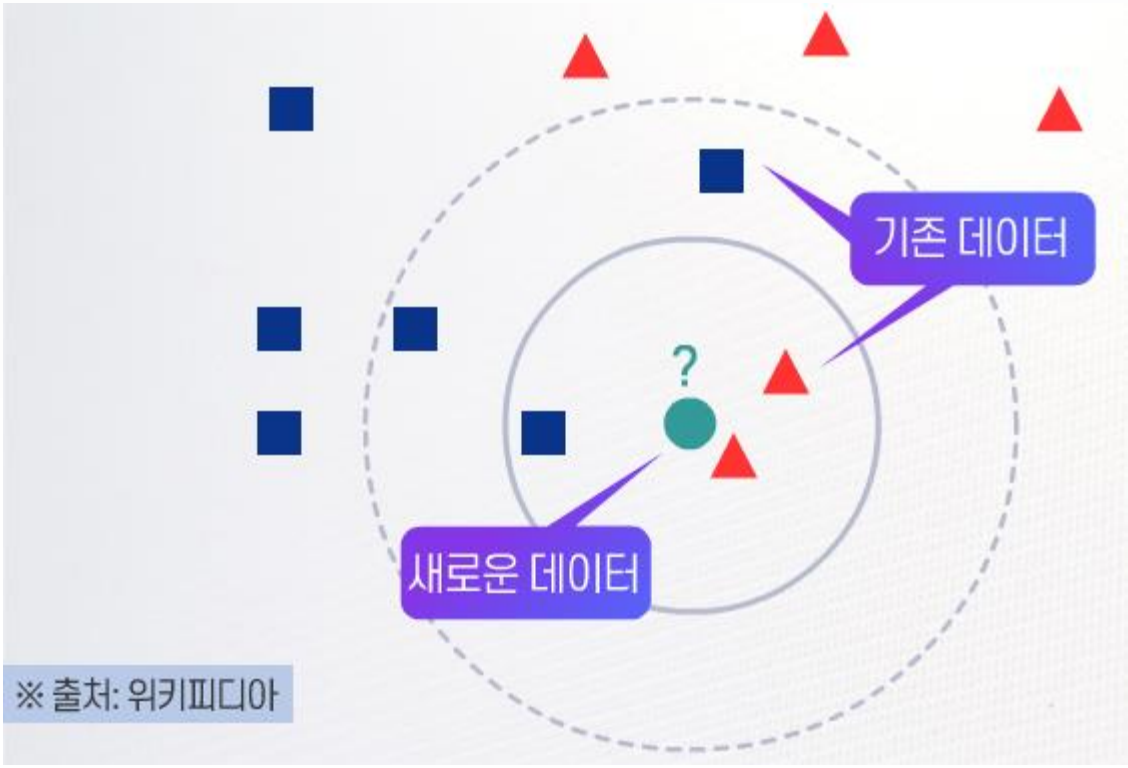
<아파트 크기에 따른 매매가격 분석>

집의 크기별 가격 데이터를 학습하여 새로운 집의 가격을 예측함

4) K-NN 알고리즘

(1) K-NN 알고리즘(K-Nearest Neighbor, K-최근접 이웃 알고리즘)

- 지도학습의 알고리즘 중 비교적 간단하며 직관적이고 이해하기 쉬움
- 분류 또는 회귀 모두 사용 가능함
- 특정 공간 내 입력된 데이터와 가장 가까운 K개의 요소를 찾아 더 많이 일치하는 곳으로 분류함



초록색 데이터는 파란색 편에 속할까? 빨간색 편에 속할까?

→ K의 개수에 따라 새롭게 들어온 데이터와 가장 가까운 데이터를 찾아 새로운 데이터가 어느 쪽에 분류되는지 판단함

4) K-NN 알고리즘

(1) K-NN 알고리즘(K-Nearest Neighbor, K-최근접 이웃 알고리즘)

(2) K-NN의 장점

- 고전적인 알고리즘인 것에 비해 높은 정확도
  - 기존 데이터 모두를 매번 새롭게 검사하기 때문임
- 오류 데이터 혹은 이상 데이터가 결과 값에 크게 영향을 미치지 않음
  - 가장 가까운 K개의 데이터를 활용하기 때문임

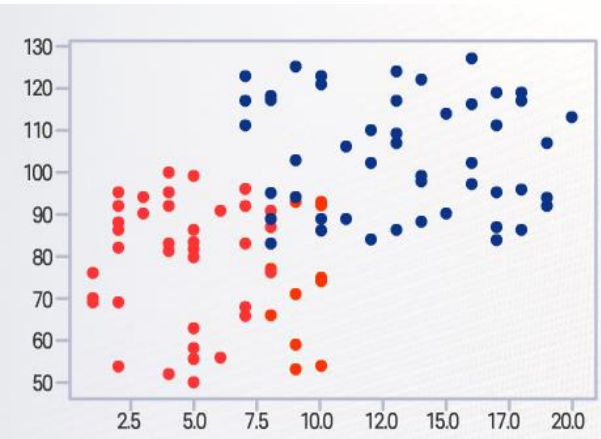
(3) K-NN의 단점

- K의 개수에 따라 결과가 달라질 수 있음
- 항상 모든 데이터를 비교해야 하므로 느림(처리 시간 증가)
- 데이터가 많으면 많을수록 더 느려짐
- 데이터가 많으면 정확도는 올라가나, 고사양의 컴퓨터 필요함 (많은 메모리 사용)

5) K-NN 알고리즘 실습 with Python

(1) Python의 Numpy와 Random module을 사용하는 이유

- 데이터를 직접 구하기 어려워 임의의 데이터를 생성하기 위해 활용함



실습 예제: 귤과 오렌지 분류

- 분류를 위한 속성 값
  - → 크기(귤: 1~10, 오렌지: 7~20)
  - → 무게(귤: 50~100, 오렌지: 80~130)
  - → 파랑: 오렌지, 빨강: 귤

(2) 임의 데이터 생성

```
import random
import numpy as np

r = [] # 귤 1
b = [] # 오렌지 0
for i in range(50):
    r.append([random.randint(1, 10),random.randint(50, 100),1])
    b.append([random.randint(7, 20),random.randint(80, 130),0])
```

5) K-NN 알고리즘 실습 with Python

(3) K-NN 알고리즘 구현

```
def distance(x,y):
    #두 점 사이의 거리를 구하는 함수
    return np.sqrt(pow((x[0]-y[0]),2)+pow((x[1]-y[1]),2))

def knn(x,y,k):
    result=[]
    cnt=0
    for i in range(len(y)):
        result.append([distance(x,y[i]),y[i][2]])
    result.sort()
    for i in range(k):
        if (result[i][1]==1):
            cnt +=1
    if (cnt > (k/2)):
        print ("이것은 귤입니다.")
    else:
        print("이것은 오렌지입니다.")
```

(4) 새로운 데이터 입력

```
weight = input("무게를 입력해주세요. ")
size = input("크기를 입력해주세요. ")
num = input("k를 입력해주세요. ")
new = [int(size),int(weight)]

knn(new,r+b,int(num))
```

```
무게를 입력해주세요. 100
크기를 입력해주세요. 15
k를 입력해주세요. 3
이것은 오렌지입니다.
```

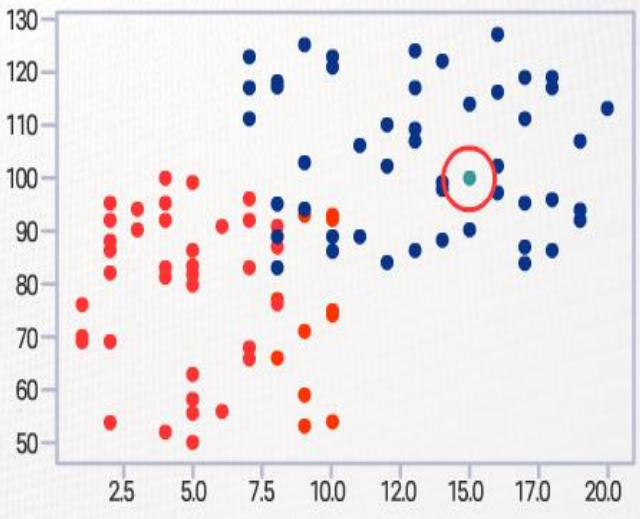
2. 지도학습

5) K-NN 알고리즘 실습 with Python

(5) 그래프로 확인

- 초록색: 새로운 데이터

```
import matplotlib.pyplot as plt
%matplotlib inline
rr = np.array(r)
bb = np.array(b)
for i,j in rr[:,:2]:
    plt.plot(i,j,'or')
for i,j in bb[:,:2]:
    plt.plot(i,j,'ob')
plt.plot(int(size),int(weight),'og')
plt.show()
```

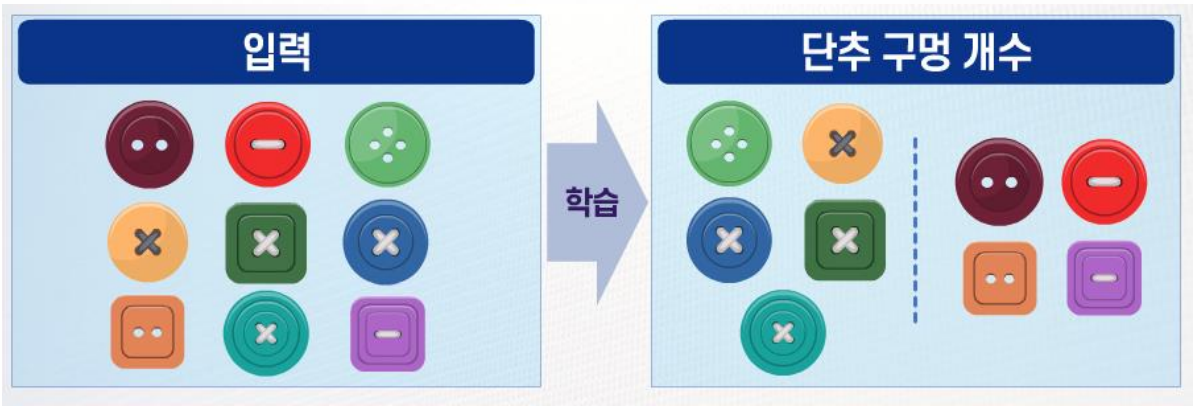


3. 비지도학습

1) 비지도학습의 정의

(1) 비지도학습(Unsupervised learning)

- 입력 데이터만 주고 컴퓨터에게 스스로 답을 찾을 수 있는 프로그램을 작성하도록 학습시키는 방법

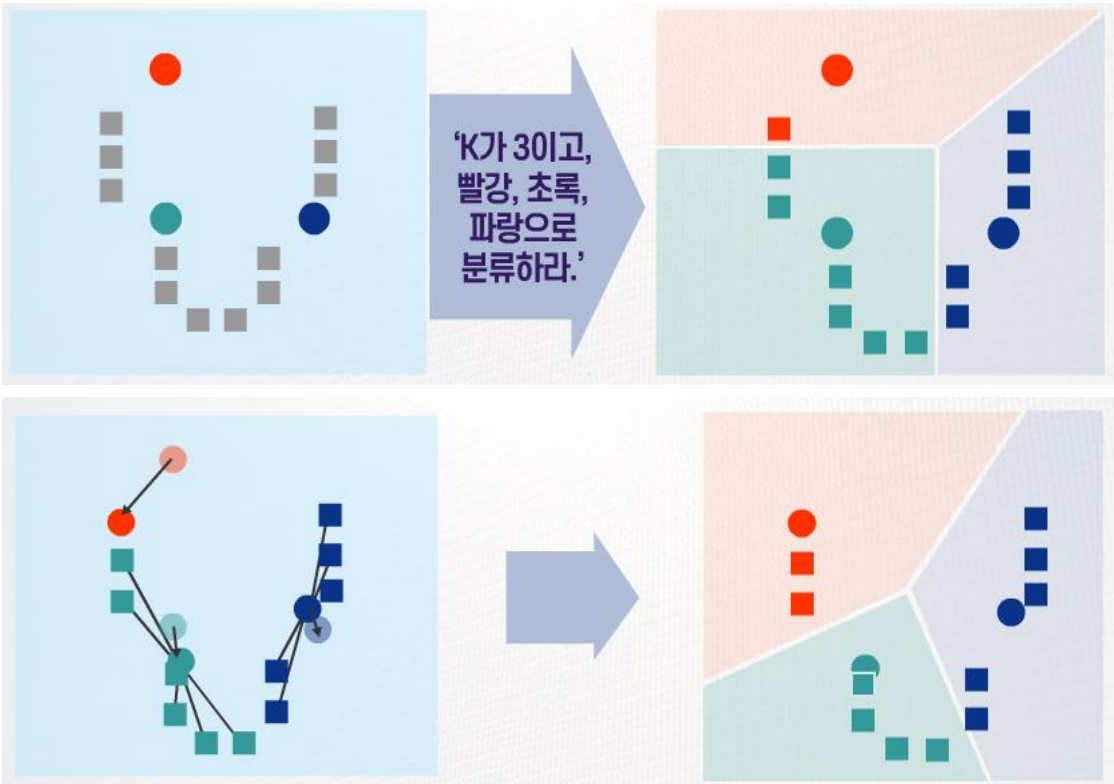


- 컴퓨터가 스스로 어떤 패턴을 찾아 군집화하여 나름대로의 정답을 만들어 나감
- 컴퓨터에게 입력 데이터에 대해 정답이 무엇인지 알려주지 않음
- 목적 값(Target Value)이 없음
- 입력된 데이터를 토대로 나름대로의 데이터의 구성을 알아내려고 함
  - 데이터의 패턴, 구성, 특징 등을 스스로 파악함
  - 클러스터링(Clustering, 데이터 군집화) 방법

2) K-Means 알고리즘

(1) K-Means 알고리즘(K-평균 알고리즘)

- 비지도학습의 군집화 방법 중 하나임
  - 입력된 데이터를 K개의 클러스터(Cluster)로 묶는 방법
- 입력 데이터를 K개의 클러스터(Cluster)로 묶는 방법
  - K개의 중심점을 잡고 입력데이터를 분류하고 무게중심을 기준으로 중심점을 이동하여 군집화 함



※ 출처 : 위키피디아

- 중심점이 더 이상 변하지 않을 때 학습을 종료시키는 방식으로 알고리즘이 진행됨

2) K-Means 알고리즘

(2) K-Means 알고리즘의 장점

- 간단한 알고리즘이므로 다른 알고리즘에 비해 쉽게 구현이 가능함
- 입력 데이터의 정답이 필요 없음

  - 주어진 데이터의 사전 정보 없이 의미 있는 결과를 찾아낼 수 있음

(3) K-Means 알고리즘의 단점

- K를 직접 정해주어야 하고, K의 개수에 따라 결과 값이 완전히 달라짐
- 오류 데이터, 이상 데이터에 민감함

  - 데이터들의 평균을 구해야 하기 때문임
- 결과 해석이 어려움

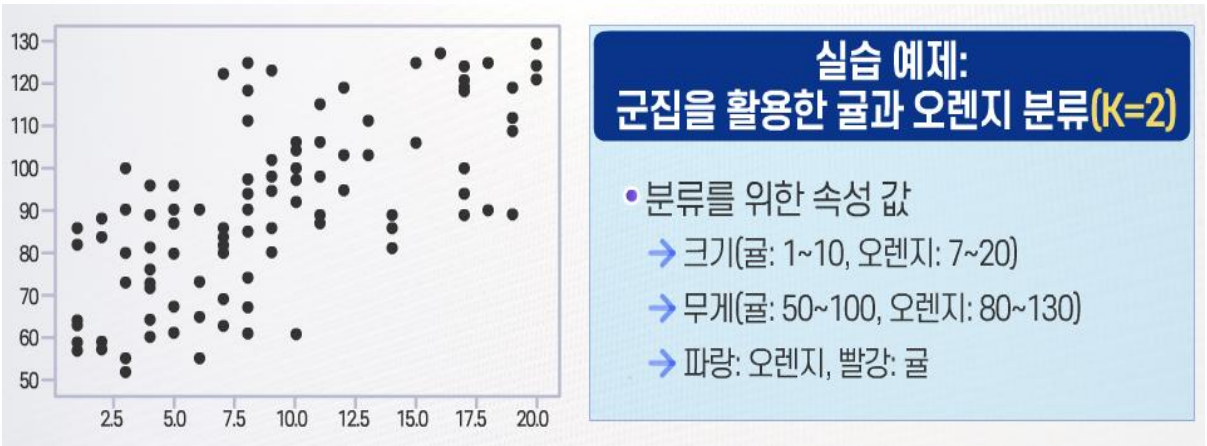
  - K의 개수 및 각각의 해석에 따라 결과가 달라질 수 있음

3. 비지도학습

3) K-Means 알고리즘 실습 with Python

(1) Python의 Random modul 사용

- K-NN 알고리즘과 동일하게 데이터를 직접 구하기 어려워 사용
- 임의의 데이터 사용을 위해 Random module 사용



3) K-Means 알고리즘 실습 with Python

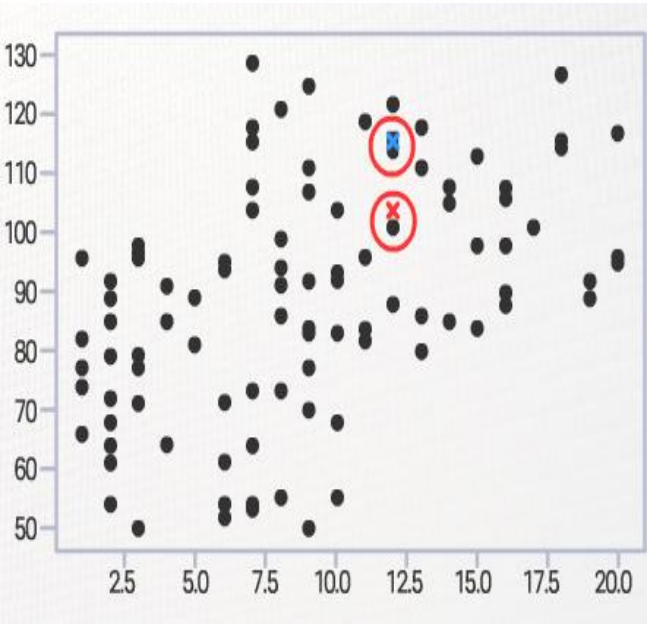
(2) 임의 데이터 및 초기 중심점 생성

```
import random
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

data = []
for i in range(50):
    data.append([random.randint(1, 10),random.randint(50, 100)])
    data.append([random.randint(7, 20),random.randint(80, 130)])

#초기 랜덤 값 2개
random_points = [[random.randint(1, 20),random.randint(50, 130)],
                 [random.randint(1, 20),random.randint(50, 130)]]

#데이터와 랜덤 값 그래프
for i in data:
    plt.plot(i[0],i[1],'o',color='k')
    plt.plot(random_points[0][0],random_points[0][1],'x',color='r')
    plt.plot(random_points[1][0],random_points[1][1],'x',color='b')
```



3) K-Means 알고리즘 실습 with Python

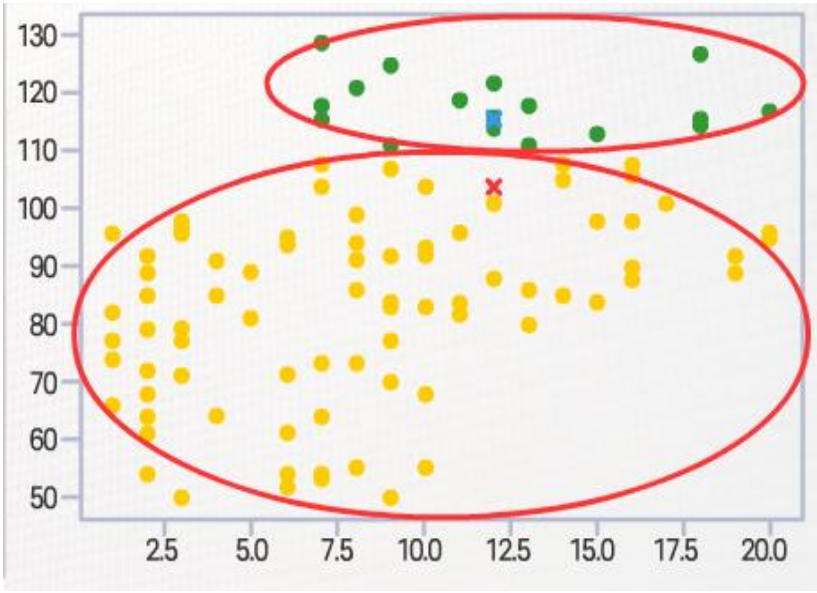
(3) 중심점을 기준으로 영역 분리

```
#두 영역을 나눌 빈 리스트 생성
tmp1 = []
tmp2 = []

#영역을 나누기 위해 두 점 사이의 거리를 구하는 함수
def dist(x,y):
    return np.sqrt((x[0]-y[0])**2 +(x[1]-y[1])**2)

#각 랜덤 점과 모든 점들의 거리를 구해 가까운 쪽의 영역으로 추가
for i in data:
    if (dist(random_points[0],i) > dist(random_points[1],i)):
        tmp2.append(i)
    else:
        tmp1.append(i)

for i in tmp1:
    plt.plot(i[0],i[1],'o',color='y')
for i in tmp2:
    plt.plot(i[0],i[1],'o',color='g')
    plt.plot(random_points[0][0],random_points[0][1],'x',color='r')
    plt.plot(random_points[1][0],random_points[1][1],'x',color='b')
```



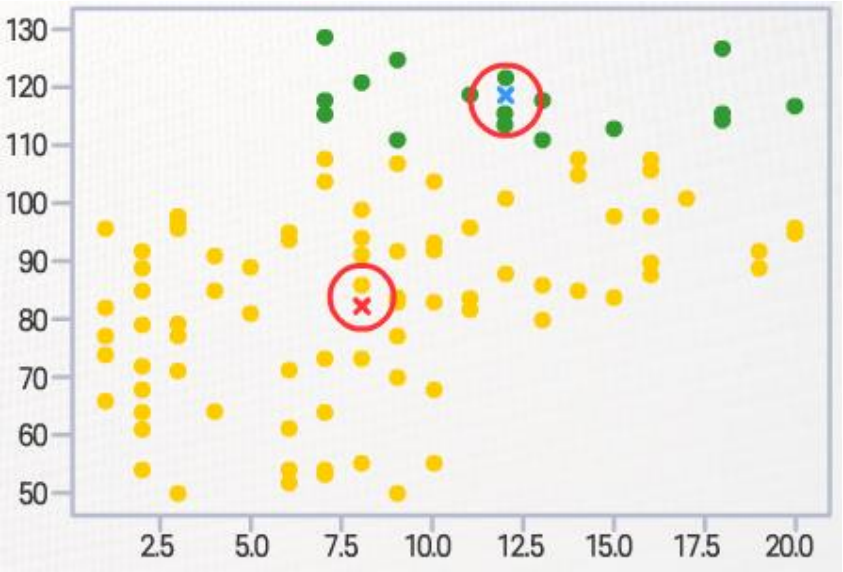
3) K-Means 알고리즘 실습 with Python

(4) 새로운 중심점으로 이동

```
# 점 이동
sum1=0
sum2=0
for i in tmp1:
    sum1 +=i[0]
    sum2 +=i[1]

new_points = []
new_points.append([sum1/len(tmp1),sum2/len(tmp1)])
sum1=0
sum2=0
for i in tmp2:
    sum1 +=i[0]
    sum2 +=i[1]
new_points.append([sum1/len(tmp2),sum2/len(tmp2)])

#새로운 점 그래프
for i in tmp1:
    plt.plot(i[0],i[1],'o',color='y')
for i in tmp2:
    plt.plot(i[0],i[1],'o',color='g')
plt.plot(new_points[0][0],new_points[0][1],'x',color='r')
plt.plot(new_points[1][0],new_points[1][1],'x',color='b')
```



3) K-Means 알고리즘 실습 with Python

(5) 더 이상 중심 이동이 없을 때까지 반복

※ 해당 예제는 데이터가 단순하고 경험적으로 10회 정도만 반복해도
더 이상 중심점 이동이 없지만, 데이터에 따라 그 횟수는 다를 수 있습니다.

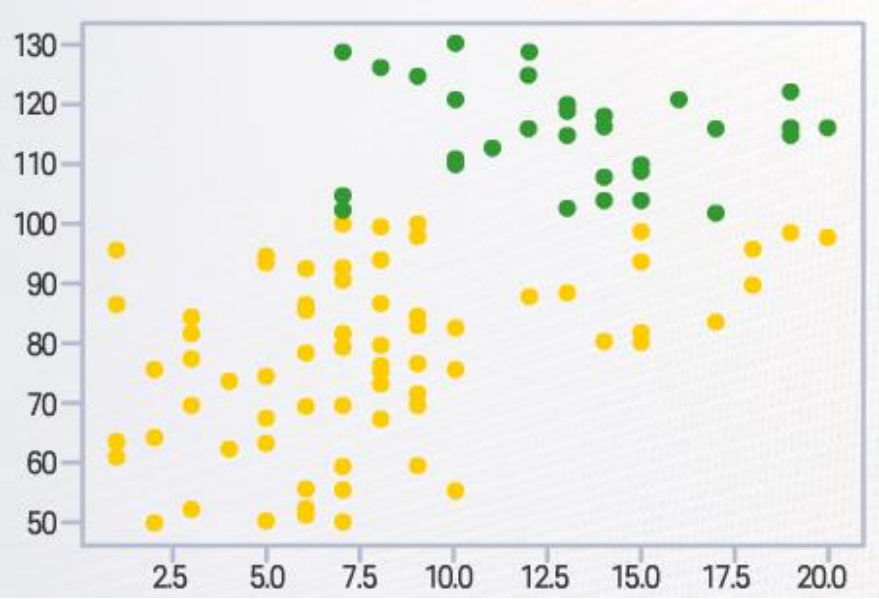
```
#전체코드
def dist(x,y):
    return np.sqrt((x[0]-y[0])**2 +(x[1]-y[1])**2)

data = []
for i in range(50):
    data.append([random.randint(1, 10),random.randint(50, 100)])
    data.append([random.randint(7, 20),random.randint(80, 130)])
random_points = [[random.randint(1, 20),random.randint(50, 130)],
                 [random.randint(1, 20),random.randint(50, 130)]]

for k in range(10):
    tmp1 = []
    tmp2 = []
    for i in data:
        if (dist(random_points[0],i) > dist(random_points[1],i)):
            tmp2.append(i)
        else:
            tmp1.append(i)
    sum1=0
    sum2=0
    for i in tmp1:
        sum1 +=i[0]
        sum2 +=i[1]
    new_points = []
    new_points.append([sum1/len(tmp1),sum2/len(tmp1)])
    sum1=0
    sum2=0
    for i in tmp2:
        sum1 +=i[0]
        sum2 +=i[1]
    new_points.append([sum1/len(tmp2),sum2/len(tmp2)])
```

3) K-Means 알고리즘 실습 with Python

(5) 더 이상 중심 이동이 없을 때까지 반복



1. 기계학습

- 인공지능의 방법 중 하나임
- 딥러닝 또한 기계학습의 한 종류임
- 컴퓨터에게 명시적으로 프로그램을 하지 않고도 컴퓨터가 학습을 알 수 있는 능력을 갖게 하는 것임
- 지도학습과 비지도학습으로 구분되며, 정답 데이터의 유무에 따라 분류함

2. 지도학습

- 컴퓨터에게 입력과 정답 데이터 모두를 알려주고, 컴퓨터가 데이터의 특징을 파악할 수 있도록 학습하는 방법임
- 분류와 예측(회귀) 두 가지 방법이 있으며, 데이터의 종류 (주로 연속성)로 분류 가능함
- 'K-NN'이 대표적인 알고리즘 (새로운 데이터와 가장 가까운 K개의 개수로 데이터 분류)
  - 매번 모든 데이터를 계산해야 하므로 메모리도 많이 필요하고 느림
  - 비교적 정확한 분류 가능함

3. 비지도학습

- 컴퓨터에게 정답을 알려주지 않고 스스로 특징 및 패턴 등을 찾을 수 있도록 학습하는 방법임
- K-Means가 대표적 알고리즘임

  (데이터의 구조를 찾아 군집화하는 방법)

  - K의 개수에 따라 결과가 많이 달라짐
  - 다른 알고리즘에 비해 쉽게 구현 가능함