Euron_ML_Week13

7. 군집화

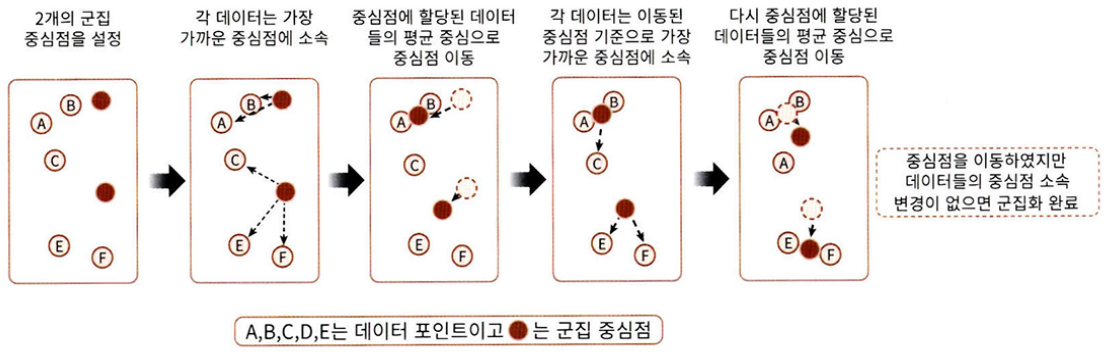
01. K-평균 알고리즘 이해

📎 K-평균

: 군집화에서 가장 일반적으로 사용되는 알고리즘

: 군집 중심점(centroid)이라는 특정한 임의의 지점을 선택해 해당 중심에 가장 가까운 포인트들을 선택하는 군집화 기법

> 군집중심점: 모든 데이터 포인트에서 더이상 중심점의 이동이 없을 경우에 반복을 멈추고 해당 중심점에 속하는 데이터 포인트들을 군집화
>
> 선택된 포인트의 평균 지점으로 이동하고 이동된 중심점에서 다시 가까운 포인트를 선택, 다시 중심점을 평균 지점으로 이동하는 프로세스를 반복적으로 수행



- K-평균의 장점

  : 일반적인 군집화에서 가장 많이 활용되는 알고리즘

  : 쉽고 간결한 알고리즘

- K-평균의 단점

  : 속성의 개수가 많을 경우 군집화 정확도가 떨어짐

  : 반복 횟수가 많을 경우 수행 시간이 느려짐

  : 몇 개의 군집을 선택할지 가이드하기 어려움

- 사이킷런 KMeans 클래스 소개

- 하이퍼 파라미터

n_clusters	군집화할 개수(군집 중심점의 개수)
init	초기에 군집 중심점의 좌표를 설정할 방식. 보통은 임의로 설정하지 않고 K-Means++ 방식으로 최초 설정
max_iter	최대 반복 횟수. 이 횟수 이전에 모든 데이터의 중심점 이동이 없으면 종료

- 속성

labels_	각 데이터 포인트가 속한 군집중심점 레이블

cluster_centers_	: 각 군집 중심점 좌표(shape=[군집개수, 피처개수]). 이를 이용해 시각화 가능

- K-평균을 이용한 붓꽃 데이터 세트 군집화

colab에서 실행

- 군집화 알고리즘 테스트를 위한 데이터 생성

- 대표적인 군집화용 데이터 생성기
  - make_blobs(): 개별 군집의 중심점과 표준 편차 제어 기능이 추가
  - make_classification(): 노이즈를 포함한 데이터를 만드는 데 유용하게 사용

- make_blobs()의 사용법
  - 호출 파라미터

n_samples	생성할 총 데이터의 개수(디폴트 = 100)
n_features	데이터의 피처 개수 시각화를 목표로 할 경우 2개로 설정
centers	int 값 (ndarray형태로 표현할 경우) 개별 군집 중심점의 좌표
cluster_std	생성될 군집 데이터의 표준편차 — float로 입력: 군집 내 데이터의 표준 편차 — [float, ...]로 입력: 각 군집의 순서대로 각각의 표준편차가 만들어짐. ⇒ 군집별로 서로 다른 표준편차를 가진 데이터 세트를 만들 때 사용

colab에서 실행

---

02. 군집 평가(Cluster Evalution)

군집화가 효율적으로 잘 됐는지 평가할 수 있는 지표(군집화 성능 평가): 실루엣 분석

- 실루엣 분석의 개요

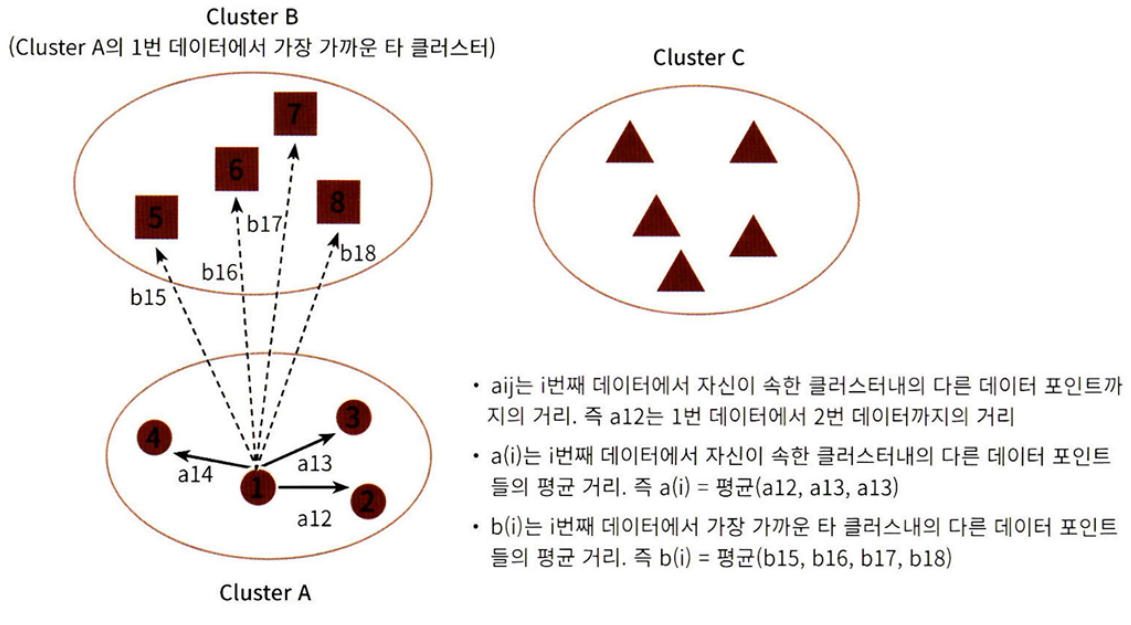
📎 실루엣 분석(silhouette analysis)

: 각 군집 간의 거리가 얼마나 효율적으로 분리돼 있는지 나타냄

> 효율적으로 분리: 다른 군집과의 거리는 떨어져 있고 동일 군집끼리의 데이터는 서로 가깝게 잘 뭉쳐 있음.
>
> (군집화가 잘 될수록 개별 군집은 비슷한 정도의 공간을 가지고 떨어져 있음)

: 실루엣 계수(silhouette coefficient)를 기반으로 함

- 실루엣 계수: 개별 데이터가 가지는 군집화 지표
  - 같은 군집 내의 데이터: 얼마나 가깝게 군집화되어 있는지
  - 다른 군집 내의 데이터: 얼마나 멀리 분리되어 있는지
  - -1에서 1 사이의 값을 가짐
    - 1로 가까워짐: 근처의 군집과 더 멀리 떨어져 있음
    - 0에 가까워짐: 근처의 군집과 가까워짐
    - - 값: 아예 다른 군집에 데이터 포인트가 할당됨



- 해당 데이터 포인트와 같은 군집 내에 있는 다른 데이터 포인트와의 평균 거리: a(i)
- 해당 데이터 포인트가 속하지 않은 군집 중 가장 가까운 군집과의 평균 거리: b(i)
- 두 군집 간의 거리가 얼마나 떨어져 있는가: b(i)- a(i)
- 정규화: (b(i)- a(i))/MAX(a(i),b(i))

$$s(i) = \frac{(b(i) - a(i))}{(\max(a(i), b(i))}$$

- 사이킷런의 실루엣 분석 메소드
  - `silhouette_sample(X, labels, metric='euclidean', **kwds)`

    : 인자로 X_feature 데이터 세트, 군집 레이블 값(labels) 입력

    ⇒ 각 데이터의 실루엣 계수를 계산하여 반환
  - `silhouette_score(X, labels, metric='euclidean', sample_size=None, **kwds)`

    : 인자로 X feature 데이터 세트, 군집 레이블 값(labels) 입력

    ⇒ 전체 데이터의 실루엣계수 값을 평균하여 반환
    - `np.mean(silhouette_samples())` 랑 같음
    - 일반적으로 이 값이 높을수록 군집화가 어느정도 잘 됐다고 판단할 수 있지만, 무조건 그런건 아니다.
- 좋은 군집화 조건
  a. 전체 실루엣 계수의 평균값(사이킷런의 silhoulette_score() 값)은 0~1 사이의 값을 가지며, 1에 가까울수록 좋음
  b. 전체 실루엣 계수의 평균값과 더불어 개별 군집의 평균값의 편차가 크지 않아야 함.
  c. 즉, 개별 군집의 실루엣 계수 평균값이 전체 실루엣 계수의 평균값에서 크게 벗어나지 않는 것이 중요

- 붓꽃 데이터 세트를 이용한 군집 평가

colab에서 실행

- 군집별 평균 실루엣 계수의 시각화를 통한 군집 개수 최적화 방법

- 전체 데이터의 평균 실루엣 계수 값이 높다고 해서, 반드시 최적의 군집 개수로 군집화가 잘 됐다고 볼 수 없음

  : 특정 군집만 실루엣 계수가 엄청 높고 나머지 군집들은 낮아도, 평균 실루엣 계수 자체는 높게 나올 수 있기 때문

---

03. 평균 이동

- 평균 이동 (Mean Shift)의 개요

📎 평균 이동

: K-평균과 유사하게 중심을 군집의 중심으로 지속적으로 움직이며 군집화 수행

: 데이터의 분포도를 이용해 군집 중심점을 찾음

> 군집중심점
>
> : 데이터 포인트가 모여있는 곳
> : 확률 밀도 함수(probability density function) 이용
> → 확률 밀도 함수가 피크인 점을 군집 중심점으로 선정
>
> : 확률 밀도 함수를 찾기 위해 KDE(Kernel Density Estimation)을 이용

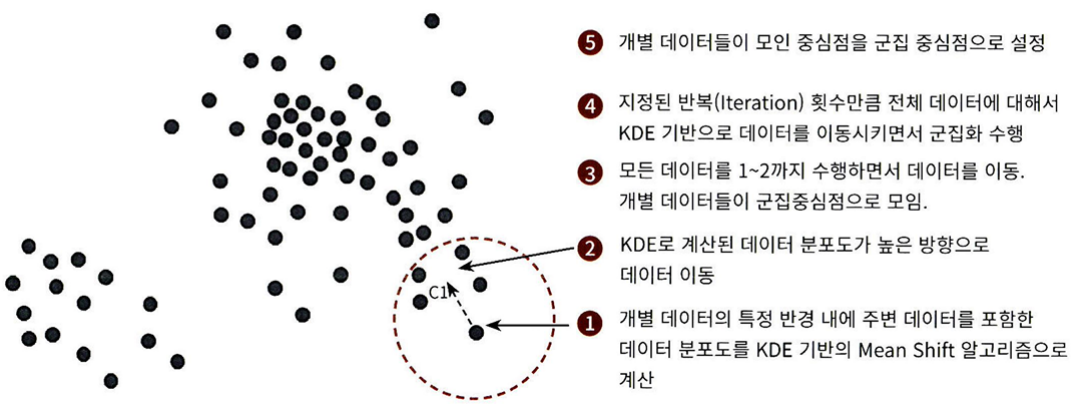
- K-평균과의 차이점
  - K-평균 : 중심에 소속된 데이터의 평균 거리 중심으로 이동
  - 평균 이동 : 중심을 데이터가 모여 있는 밀도가 가장 높은 곳으로 이동

- 이동 방식

  : 특정 데이터를 반경 내의 데이터 분포 확률 밀도가 가장 높은 곳으로 이동하기 위해 주변 데이터와의 거리 값을 KDE 함수 값으로 입력

  → 반환 값을 현재 위치에서 업데이트

  이를 반복 적용하며 데이터의 군집 중심점을 찾아냄



- 확률 밀도 함수 PDF(Probability Density Function)

  : 확률 변수의 분포를 나타내는 함수

  : 정규분포 함수, 감마 분포, t-분포 등이 존재

  : 확률 밀도 함수를 통해 특정 변수가 어떤 값을 갖게 될지에 대한 확률을 알게됨

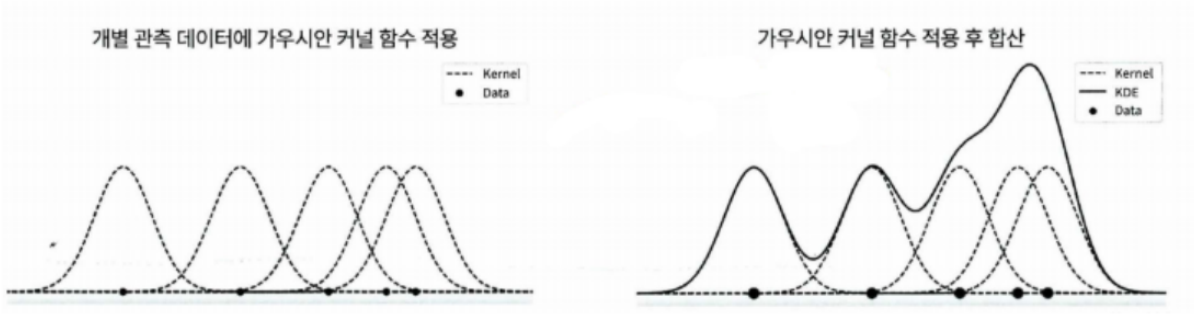
  ⇒ 변수의 특성, 확률 분포 등 변수의 많은 요소를 알 수 있음

- KDE(Kernel Density Estimation)

  : 커널(Kernel) 함수를 통해 어떤 변수의 확률 밀도 함수를 측정하는 방법

  : 개별 관측 데이터에 커널 함수를 적용한 뒤, 이 적용 값을 모두 더한 후 개별 관측 데이터의 건수로 나눠 확률 밀도 함수를 추정

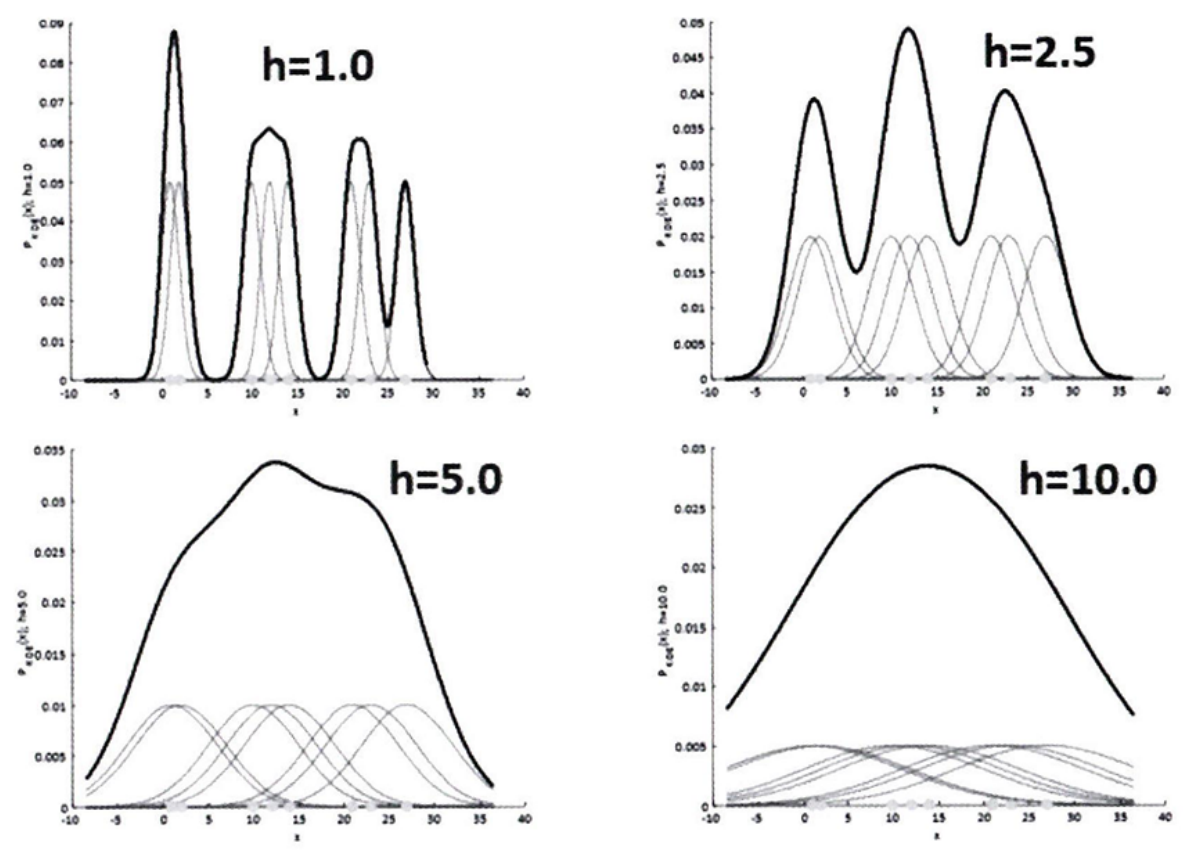
  : 대표적인 커널 함수 → 가우시안 분포 함수



$$\text{KDE} = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

: K는 커널 함수, x는 확률 변숫값, xi는 관측값, h는 대역폭(bandwidth)

: 대역폭 h는 KDE 형태를 부드러운(또는 뾰족한) 형태로 평활화(Smoothing)하는 데 적용. h를 어떻게 설정하느냐에 따라 확률 밀도 추정 성능을 좌우



- 작은 h 값 (h=1.0)

  : 좁고 뾰족한 KDE를 가짐.

  : 변동성이 큰 방식으로 확률 밀도 함수를 추정하므로 과적합하기 쉬움

- 큰 h 값 (h=10)

  : 과도하게 평활화(smoothing)된 KDE로 인해 지나치게 단순화된 방식

  : 확률 밀도 함수를 추정하므로 과소적합하기 쉬움

⇒ 적절한 KDE의 대역폭 h를 계산하는 것이 KDE 기반의 평균 이동 군집화에서 매우 중요!

✔

- 일반적으로 대역폭이 클수록 평활화된 KDE로 인해 적은 수의 군집 중심점을 가지며, 대역폭이 적을수록 많은 수의 군집 중심점을 가짐
- 평균 이동 군집화는 군집의 개수를 지정하지 않고, 오직 **대역폭**의 크기에 따라 군집화 수행

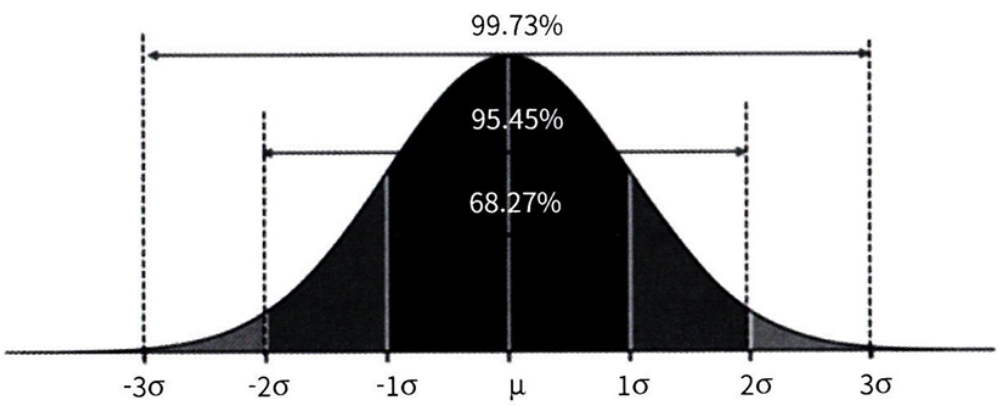
---

04. GMM(Gaussian Mixture Model)

- GMM(Gaussian Mixture Model)소개

: 군집화를 적용하고자 하는 데이터가 여러 개의 가우시안 분포(GaussianDistribution)를 가진 데이터 집합들이 섞여서 생성된 것이라는 가정 하에 군집화를 수행하는 방식

📎 가우시안 분포



: 정규 분포

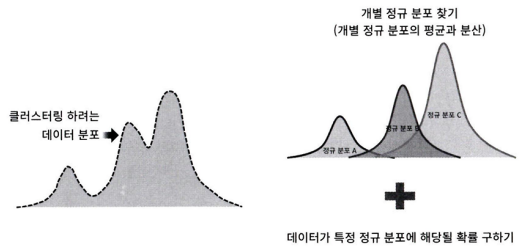
: 좌우 대칭형의 종(Bell) 형태를 가진 연속 확률 함수

: 평균 μ를 중심으로 높은 데이터 분포도를 가지고 있으며, 좌우 표준편차 1에 전체 데이터의 68.27%, 좌우 표준편차 2에 전체 데이터의 95.45%를 가짐

: 표준 정규 분포⇒ 평균이 0이고, 표준편차가 1인 정규 분포

[방식]

: 전체 데이터 세트는 서로 다른 정규 분포 형태를 가진 여러 가지 확률 분포 곡선으로 구성될 수 있으며, 이러한 서로 다른 정규 분포에 기반해 군집화를 수행



⇒ 모수추정 (이와 같은 방식을 GMM에서 모수 추정이라고 함)

: 대표적으로 2가지를 추정

- 개별 정규 분포의 평균과 분산
- 각 데이터가 어떤 정규 분포에 해당되는지의 확률

: 모수 추정을 위해 EM(Expectation and Maximization) 방법을 적용

- 사이킷런 지원

  : GaussianMixture 클래스(GMM의 EM 방식을 통한 모수 추정 군집화를 지원)

- GMM을 이용한 붓꽃 데이터 세트 군집화

colab에서 실행

- GMM과 K-평균의 비교

- KMeans
  - 개별 군집의 중심에서 원형의 범위로 군집화를 수행
  - 데이터 세트가 원형의 범위를 가질수록 KMeans의 군집화 효율 증가
  - 데이터가 원형의 범위로 퍼져 있지 않는 경우 군집화를 잘 수행하지 못함 (길쭉한 타원형으로 늘어선 경우와 같이)
- GMM
  - KMeans보다 유연하게 다양한 세트에 잘 적용될 수 있다는 장점이 있음

- 군집화를 위한 수행 시간이 오래 걸린다는 단점이 있음

[군집 시각화]

함수명: visualize_cluster_plot(clusterobj, dataframe, label_name, iscluster=True)

- 함수인자

clusterobj	사이킷런의 군집 수행 객체 KMeans나 GaussianMixture의 fit()와 predict()로 군집화를 완료한 객체 * make_blobs()로 생성한 데이터의 시각화일 경우 None 입력
dataframe	피처 데이터 세트와 label 값을 가진 DataFrame
label_name	군집화 결과 시각화일 경우 dataframe내의 군집화 label 칼럼 명 * make_blobs()결과 시각화일 경우는 dataframe내의 target 칼럼명
iscenter	사이킷런 Cluster 객체가 군집중심좌표를 제공하면 True, 그렇지 않으면 False

---

05. DBSCAN

- DBSCAN 개요

📎 DBSCAN

: 밀도 기반 군집화의 대표적 알고리즘

: 입실론 주변 영역의 최소 데이터 개수를 포함하는 밀도 기준을 충족시키는 데이터인 핵심 포인트를 연결하면서 군집화를 구성하는 방식

: 특정 공간 내에 데이터 밀도 차이를 기반 알고리즘으로 함

→ 데이터의 분포가 기하학적으로 복잡한 데이터 세트에도 효과적인 군집화 가능

- 중요 파라미터
  - 입실론 주변 영역(epsilon)

    : 개별 데이터를 중심으로 입실론 반경을 가지는 원형의 영역
  - 최소 데이터 개수(min points)

    : 개별 데이터의 입실론 주변 영역에 포함되는 타 데이터의 개수

- 데이터 포인트

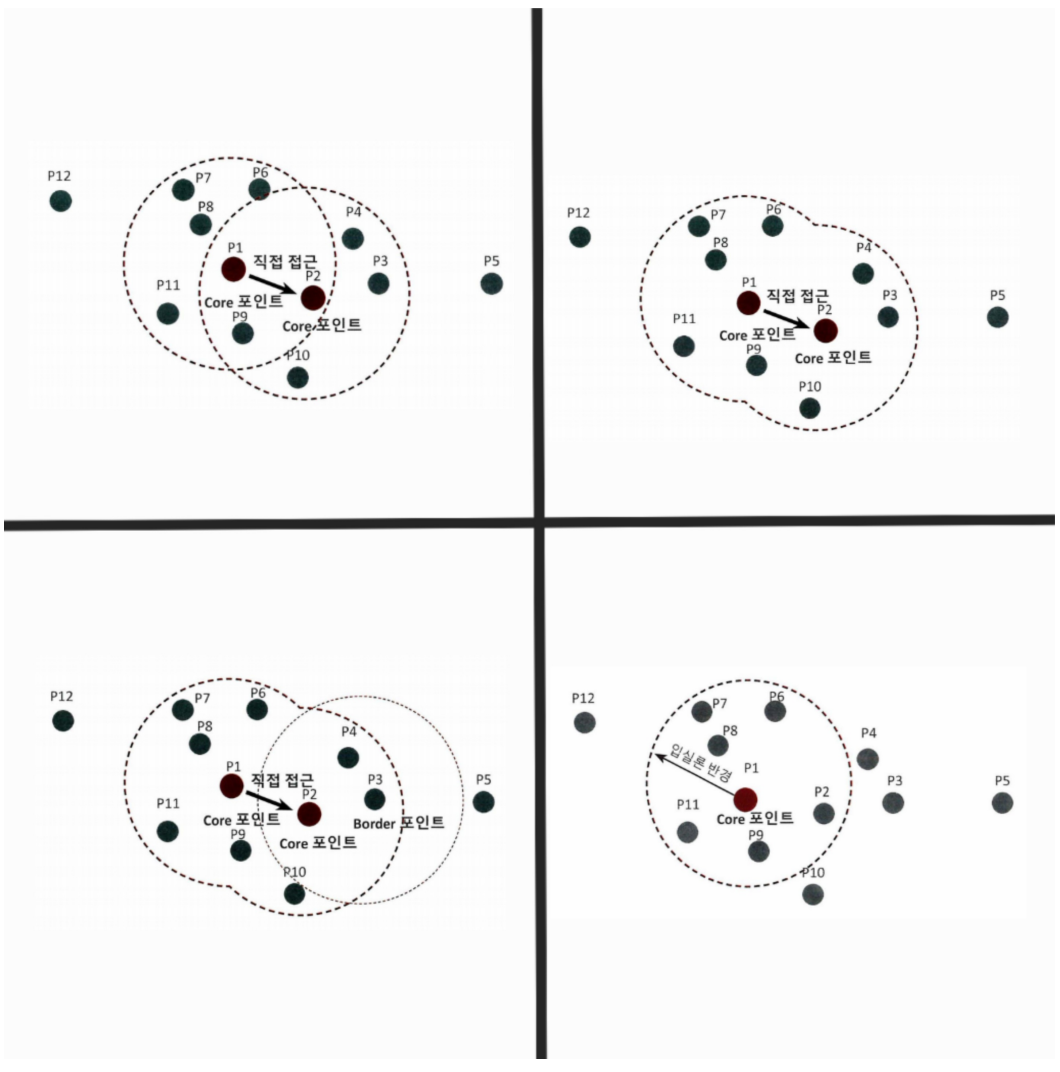
  : 입실론 주변 영역 내에 포함되는 최소 데이터 개수 충족 여부에 따라 정의
  - 핵심 포인트(Core Point)

    : 주변 영역 내에 최소 데이터 개수 이상의 타 데이터를 가지고 있는 데이터
  - 이웃 포인트(Neighbor Point)

    : 주변 영역 내에 위치한 타 데이터
  - 경계 포인트(Border Point)

    : 주변 영역 내에 최소 데이터 개수 이상의 이웃 포인트를 가지고 있지 않지만 핵심 포인트를 이웃 포인트로 가지고 있는 데이터
  - 잡음 포인트(Noise Point)

    : 최소 데이터 개수 이상의 이웃 포인트를 가지고 있지 않으며, 핵심 포인트도 이웃 포인트로 가지고 있지 않는 데이터
  - 예시



1. P1 데이터 기준 입실론 반경 내에 포함된 데이터가 최소 데이터를 만족하는 핵심 포인트
2. 핵심 포인트 P1의 이웃 데이터 포인트 P2 역시 핵심 포인트일 경우 P1에서 P2로 연결해 직접 접근이 가능
3. 특정 핵심 포인트에서 직접 접근이 가능한 다른 핵심 포인트를 서로 연결하면서 점차적으로 군집(Cluster)영역을 확장하며 군집화를 구성
4. P3의 경우 핵심 포인트가 아니지만, 이웃 데이터로 핵심 포인트를 가지고 있는 데이터인 경계 포인트. 군집의 외곽을 형성

[사이킷런 지원]

: DBSCAN 클래스

- 주요 초기화 파라미터

eps	입실론 주변 영역의 반경. 일반적으로 1 이하의 값을 설정
min_samples	핵심 포인트가 되기 위해 입실론 주변 영역 내에 포함돼야 할 데이터의 최소 개수 (자신의 데이터 포함. min points + 1)

- DBSCAN 적용하기 - 붓꽃 데이터 세트

colab에서 실행

- DBSCAN 적용하기 - make_circles()데이터 세트

📎 make_circles() 함수

: 오직 2개의 피처만을 생성

→ 별도의 피처 개수를 지정할 필요가 없음

- 파라미터
  - noise: 노이즈 데이터 세트의 비율
  - factor: 외부 원과 내부 원의 scale 비율

---

06. 군집화 실습 - 고객 세그먼테이션

- 고객 세그먼테이션의 정의와 기법

📎 고객 세그먼테이션(Customer Segmentation)

: 다양한 기준으로 고객을 분류하는 기법

: RFM 기법 이용

- RFM 기법

  : Recency(R), Frequency(F), Monetary Value(M)
  - Recency(R): 가장 최근 상품 구입 일에서 오늘까지의 기간
  - Frequency(F): 상품 구매 횟수
  - Monetary Value(M): 총 구매 금액

- 데이터 세트 로딩과 데이터 클렌징

colab에서 실행

- RFM 기반 데이터 가공

colab에서 실행

- RFM 기반 고객 세그먼테이션