ch9. 비지도 학습

https://data-scientist-brian-kim.tistory.com/76

📖 비지도 학습

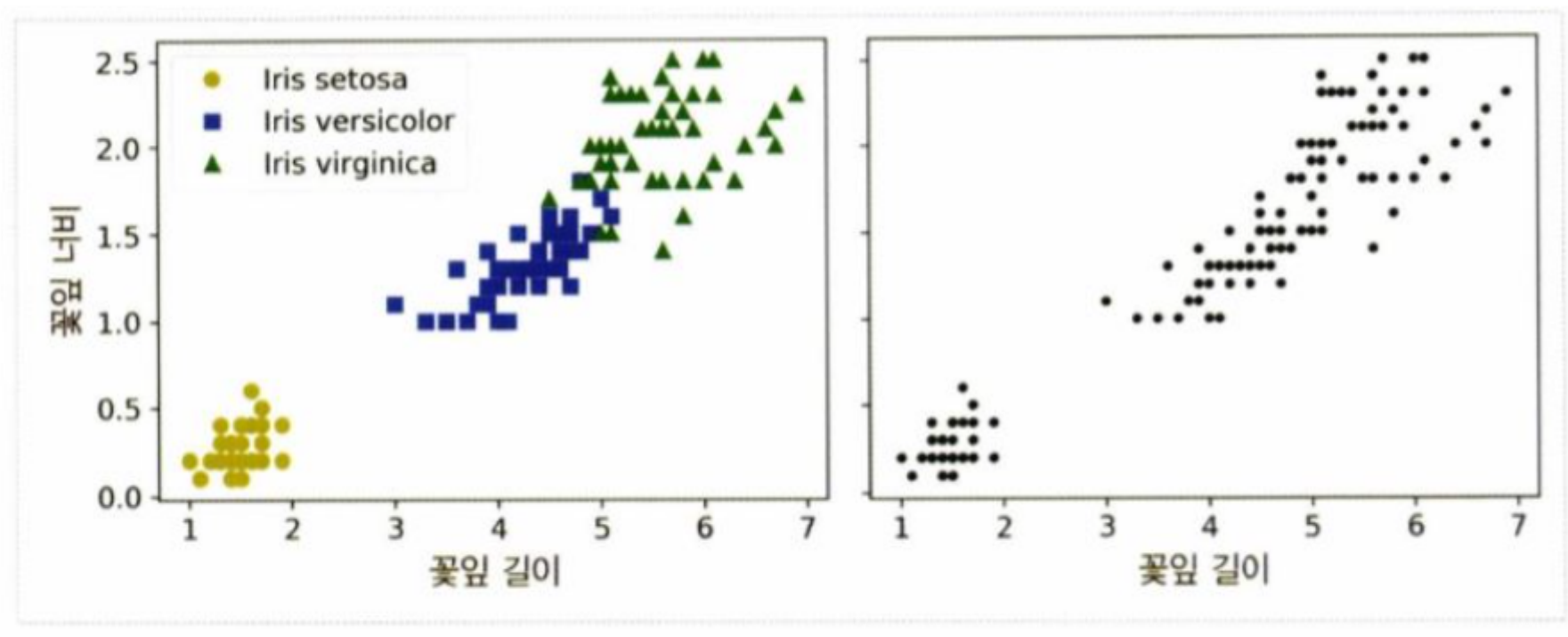
- 우리가 사용할 수 있는 **대부분의 데이터는 레이블이 없다.**
- 즉, **정답이 없는 데이터 이기** 때문에 **지도 학습을 사용할 수가 없다.**
- 이러한 경우에 사용하는 방법이 바로 **"비지도 학습"**이다. **➔ 레이블이 없는 데이터를 바로 사용하기 위한 비지도 학습**
  - 대표적인 비지도 학습으로는 **Clustering(군집 분석)**이 있다.

📖 군집분석

군집 분석은 대표적인 비지도 학습이다. 주의할 점은 "분류"와 "군집"을 헷갈리면 안 된다는 것이다.

분류
- 데이터셋이 레이블 되어 있음
- 로지스틱 회귀, SVM, 랜덤 포레스트 분류기 같은 분류 알고리즘이 잘 맞음

군집
- 레이블이 없음
- 대부분의 군집 알고리즘은 왼쪽 아래 클러스터를 쉽게 감지하지만, 오른쪽 위의 클러스터는 두 개의 하위 클러스터로 구성되었는지 확실하지 않음
- 모든 특성을 사용하면, 클러스터 세 개를 잘 구분할 수 있음



분류(왼쪽) vs 군집(오른쪽)

군집 분석은 다방면에서 활용될 수 있다.

- **고객 분류**
- **데이터 분석 → 각 cluster별로 따로 분석!!**
- **차원 축소 →** 각 클러스터에 대한 샘플의 친화성을 측정해 각 샘플의 특성 벡터 x를 클러스터 친화성의 벡터로 바꿈. k개의 클러스터가 있다면 이 벡터는 k차원이 되고, 원본 특성보다 훨씬 저차원이며 충분한 정보를 가짐. 지도 학습 알고리즘을 적용하기 전에 전처리 단계에서 활용 가능
- **이상치 탐지 →** 모든 클러스터에 친화성이 낮은 샘플은 이상치일 가능성이 높음 (결함 감지, 부정거래 감지 등)
- **준지도 학습 →** 일부 레이블된 샘플을 군집에 수행해 모든 샘플에 레이블을 전파할 수 있음. 레이블이 없는 데이터가 많고, 레이블이 있는 데이터는 적은 경우에 사용
- **검색 엔진** → 제시된 이미지와 비슷한 이미지를 찾아줌. 비슷한 이미지는 동일한 클러스터에 속함
- **이미지 분할 →** 이미지를 여러 개의 segment로 분할하는 작업. 색을 기반으로 픽셀을 클러스터로 모아 해당 클러스터의 평균 색으로 대체하여 이미지에 있는 색상의 종류를 크게 줄인다. (물제 탐지 및 추적시스템에 활용)

Cluster에 대한 보편적인 정의는 따로 없으며, 군집 알고리즘에 따라 다른 종류의 cluster를 감지한다.

▼ 1. K-means 알고리즘

- 레이블이 없는 데이터셋 샘플 중 덩어리로 잘 묶여지는 듯한 종류의 데이터셋을 빠르고 효율적으로 클러스터로 묶을 수 있는 알고리즘
- 이 알고리즘은 **각 cluster의 중심을 찾고 가장 가까운 cluster에 샘플을 할당**한다. **사이킷런의 KMeans 클래스**를 사용하면 된다. 다만 한 가지 고려해야 할 부분은 **알고리즘이 찾을 cluster 개수 k를 직접 지정해주어야 한다**는 점이다. 적절한 cluster 개수를 지정해주는 것이 쉬운 일이 아니다.
- 군집에서 각 샘플의 레이블은 알고리즘이 샘플에 할당한 클러스터의 인덱스이다.
- 일반적으로 K-means 알고리즘은 속도가 빠르다.
- 고차원 데이터 셋을 저차원으로 변환할 때, 매우 효율적인 비선형 차원 축소 기법이다.
- K-means 알고리즘은 **cluster의 크기가 많이 다르면 잘 작동하지 않는다.**
    - 샘플을 cluster에 할당할 때, centroid까지 거리를 고려해주는 것이 전부이기 때문이다.

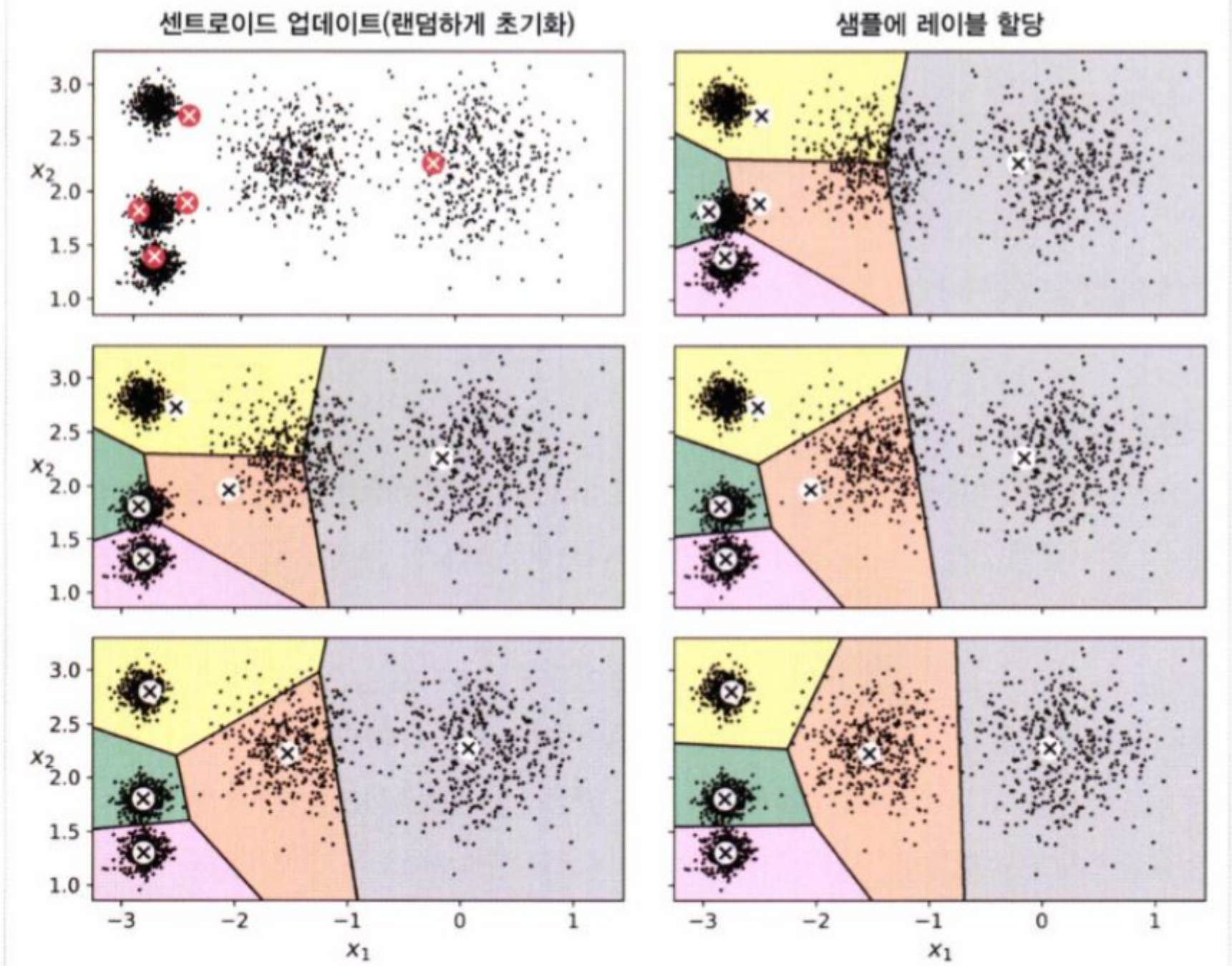
```
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

https://kdeon.tistory.com/34 → 코드확인

- 하드군집 : 샘플을 하나의 클러스터에 할당하는 것
- 소프트군집 : 클러스터마다 샘플에 점수를 부여함

<K-means 알고리즘의 작동 원리>

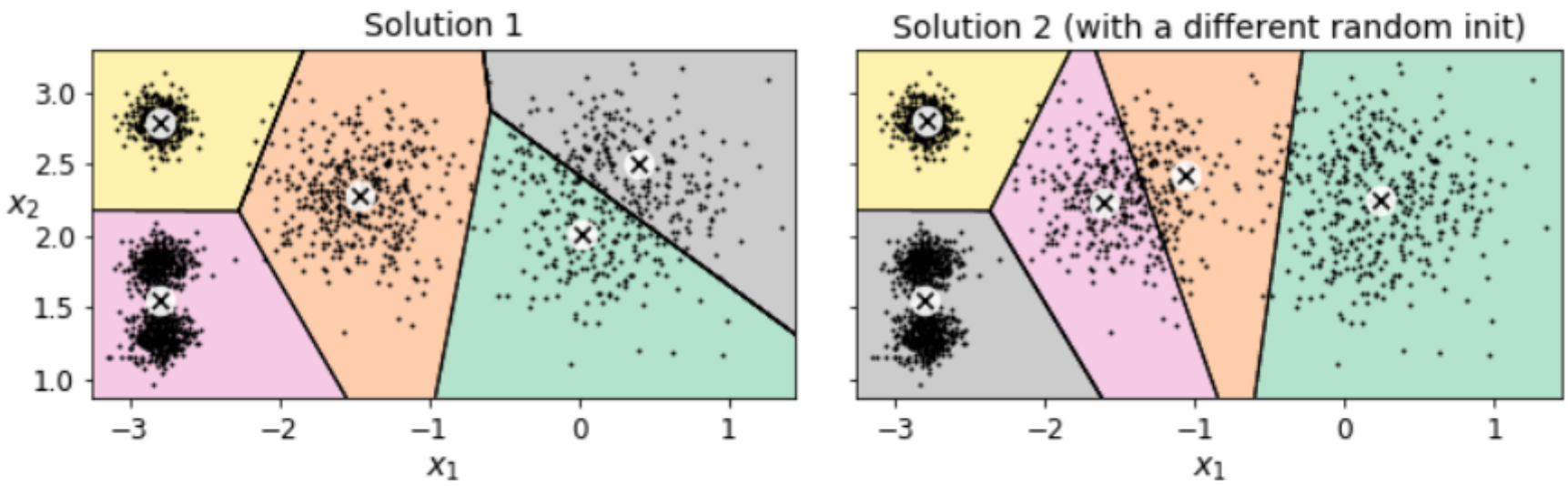
1. 레이블이나 센트로이드가 주어지지 않으면 Centroid를 랜덤하게 초기화
2. 샘플에 레이블을 할당
3. Centroid 업데이트
4. 샘플에 다시 레이블을 할당
5. 최적으로 보이는 cluster에 도달할 때까지 위 과정들을 반복!!



K-means 알고리즘의 작동 원리

이 알고리즘이 수렴하는 것이 보장되지만, 적절한 솔루션으로 수렴하지 못할 수 있음

- 이는 센트로이드 초기화에 달려 있음 (운에 따라 다름)



운 나쁜 센트로이드 초기화 때문에 만들어진 최적이 아닌 솔루션

[Centroid 초기화 방법]

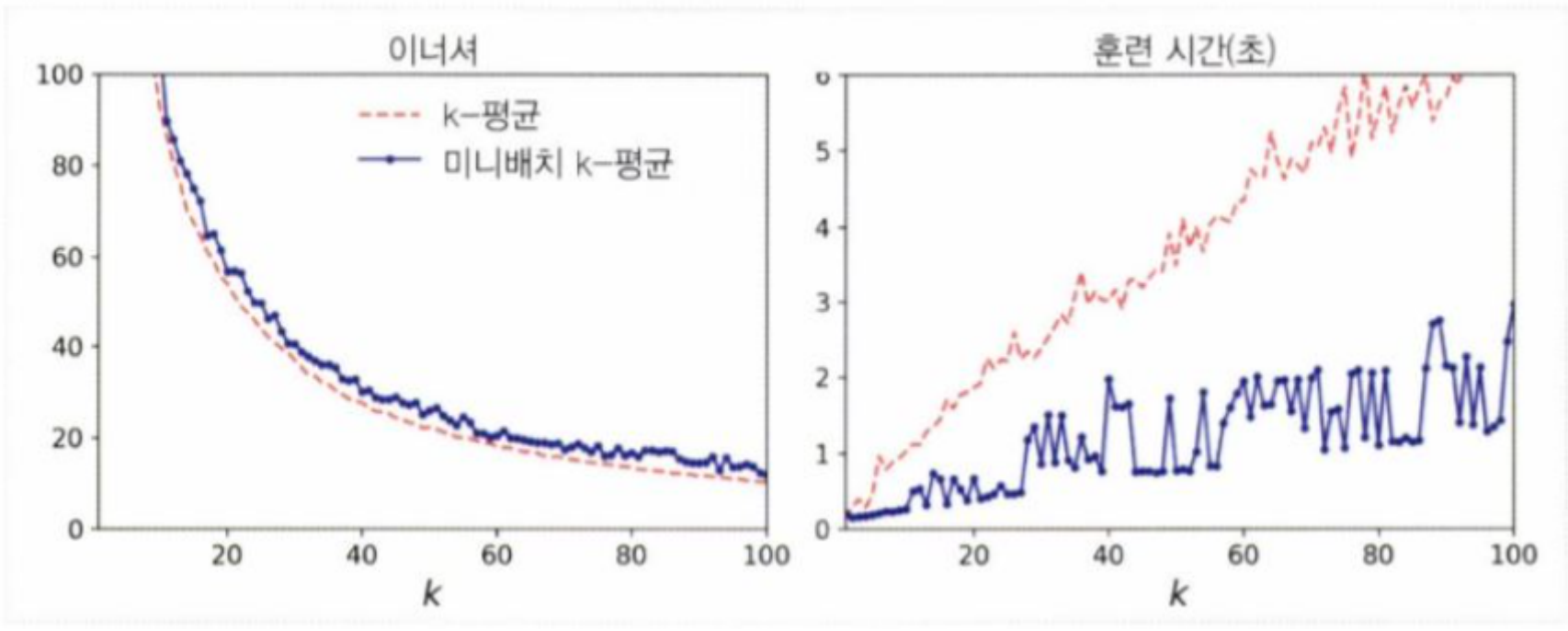
- 한 가지 방법으로는 다른 군집 알고리즘을 먼저 실행해서 centroid 위치를 근사하게 파악해 볼 수 있다.

- 또 다른 방법으로는 centroid 랜덤 초기화를 다르게 하여 여러 번 알고리즘을 실행하고, 최적의 솔루션을 선택하는 것이다.
  - 그렇다면 최적의 솔루션을 어떻게 알 수 있을까?
    - 이 때 사용하는 **성능 지표**가 바로 **모델의 이너셔(inertia)**이다.
    - **모델의 이너셔(inertia)**란 **각 샘플과 가장 가까운 centroid 사이의 평균 제곱 거리**를 의미한다.
    - **이너셔(inertia)가 낮을수록 좋은 모델**이다.

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

[미니배치 k-means 알고리즘]

- **전체 데이터 셋을 사용해 반복하지 않고, 각 반복마다 미니배치를 사용해서 centroid를 조금씩 이동**하는 방법이다.
- 이 알고리즘을 사용하면, 일반적으로 **k-means 알고리즘의 속도를 3배에서 4배 정도 높일 수 있다.**
  - 다만, **이너셔(inertia)는 일반적으로 k-means 알고리즘보다 안 좋다.**
  - **특히 cluster의 개수가 증가할 때** 위와 같은 현상이 발생한다.
- **사이킷런의 MiniBatchKMeans 클래스**를 사용하면 된다.



[최적의 cluster 개수 찾기]

- 일반적으로 최적의 cluster 개수(k)를 지정해주는 것은 쉬운 일이 아니다.
  - **왜죠? 가장 작은 이너셔(inertia)를 가진 모델을 선택하면 되는 거 아닌가요? → No!!**
  - Cluster가 늘어날수록 각 샘플은 가까운 centroid에 더 가깝게 된다. 따라서 이너셔(inertia)는 더 작아질 것이다!
- 그러면 우리는 어떻게 최적의 cluster 개수를 찾아서 지정해줄 수 있을까?

일반적으로 많이 사용하는 두 가지 방법은 다음과 같다.

1. 이너셔 그래프를 cluster 개수(k)의 함수로 그렸을 때, 그래프가 꺽이는 지점(Elbow) 찾기

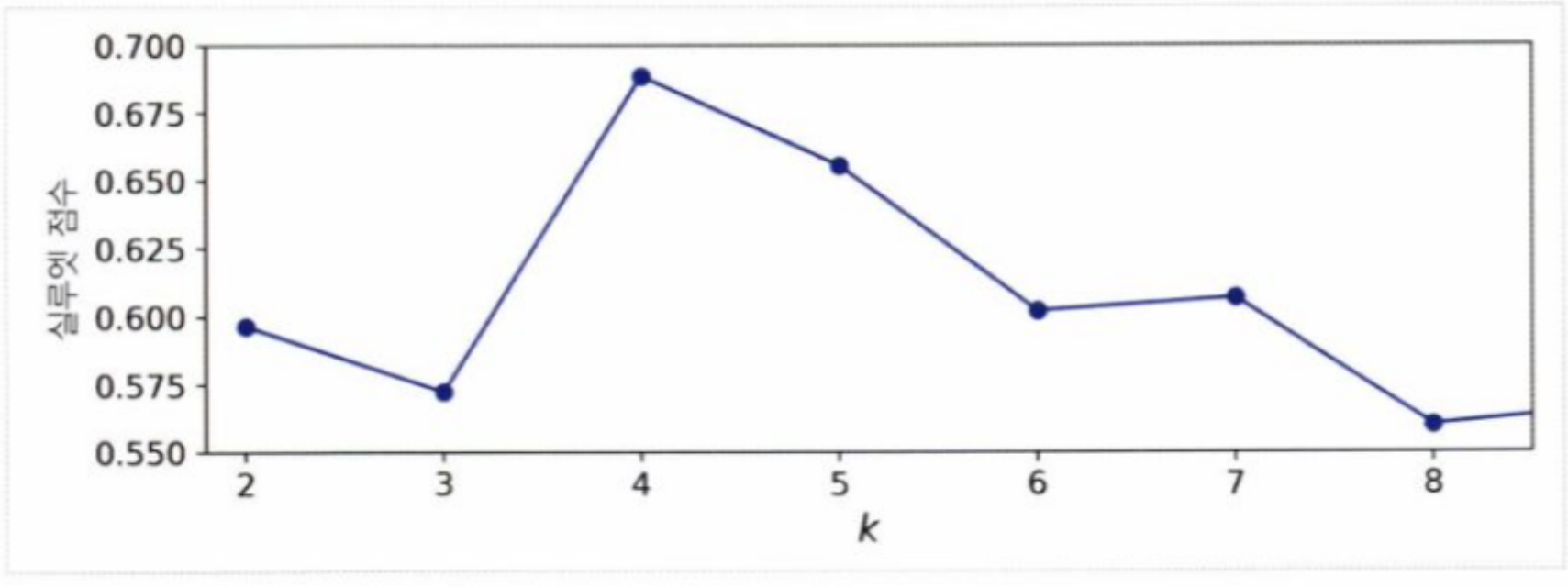
- 아래 그래프를 보면, k가 4까지 증가할 때 이너셔는 빠르게 감소한다.
- 하지만 k가 계속 증가할수록 이너셔의 감소 폭은 크게 줄어드는 것을 알 수 있다.
- 따라서 이 **그래프가 꺽이는 지점**. 즉, **Elbow가 위치한 k 값을 최적의 cluster 개수로 지정**해주면 된다.

2. 실루엣 점수 및 실루엣 다이어그램

- 물론 위의 Elbow 방법도 많이 사용하지만, 다소 주먹구구 식이라는 느낌이 든다.
- 그리하여 나온 방법이 바로 **"실루엣 점수"**이다.
  - 이 값은 **모든 샘플에 대한 실루엣 계수의 평균 값**이다.
  - **샘플의 실루엣 계수**는 **"(b - a) / max(a, b)"** 로 계산된다.
    - a : 동일한 cluster에 있는 다른 샘플까지의 평균 거리 (즉, cluster 내부의 평균 거리)
    - b : 가장 가까운 cluster까지의 평균 거리 (즉, 가장 가까운 cluster의 샘플까지 평균 거리)
  - **실루엣 계수**는 **1에서 +1까지의 값**을 가진다.
    - **"+1"**에 가까운 경우 : 해당 샘플이 cluster 안에 잘 속해 있고, 다른 cluster와는 멀리 떨어져 있다는 의미
    - **"0"**에 가까운 경우 : Cluster 경계에 위치한다는 의미
    - **"-1"**에 가까운 경우 : 해당 샘플이 잘못된 cluster에 할당되었다는 의미

- 실루엣 점수를 계산하려면 **사이킷런의 silhouette_score( ) 함수**를 사용하면 된다.

```
>>> from sklearn.metrics import silhouette_score
>>> silhouette_score(X, kmeans.labels_)
0.655517642572828
```



위 그래프를 보면 최적의 cluster 개수(k)가 4임을 알 수 있다.

<K-means 알고리즘의 한계>

- K-means 알고리즘은 속도가 빠르고 확장이 용이하다는 장점을 갖고 있다.
- 하지만 최적의 cluster 개수(k)를 직접 지정해줘야 한다는 점, 그리고 최적의 솔루션에 도달하지 못하는 문제를 해결하기 위해 알고리즘을 여러 번 실행해주어야 한다는 점이 꽤나 번거롭다.
- 또한 k-means 알고리즘은 cluster의 크기나 밀집도가 서로 다르거나, 원형이 아닐 경우(타원형 등과 같은 경우) 잘 작동하지 않는다.
  - 참고로 **타원형 cluster**에서는 **가우시안 혼합 모델(GMM)이 잘 작동**한다.
- 마지막으로 **k-means 알고리즘을 실행하기 전**에 하다.

  입력 특성의 스케일을 맞춰주는 것은 굉장히 중요

▼ 2. DBSCAN

- **밀도 기반 군집화**의 대표적인 알고리즘이다.
- **<장점>**
  - 데이터의 분포가 기하학적으로 **복잡한 데이터 셋에도 효과적**인 군집화가 가능하다.
    - 즉, **cluster의 모양과 개수에 상관없이 감지할 수 있는 능력이 뛰어나다.**
    - 따라서 **이상치에 안정적**이다.
  - 사전에 cluster 개수를 지정할 필요가 없다.
- **<단점>**
  - 데이터 밀도가 자주 변하거나, 아예 모든 데이터의 밀도가 크게 변하지 않으면 성능이 떨어진다.
  - 특성(feature)의 개수가 많으면 성능이 떨어진다.
  - **새로운 샘플에 대해 cluster를 예측할 수 없다.**
    - 따라서 사용자가 필요한 예측기를 선택해야 한다.
- DBSCAN을 구성하는 가장 **중요한 두 가지 파라미터**는 다음과 같다.
  - **입실론 주변 영역(epsilon)** : 개별 데이터를 중심으로 입실론 반경을 가지는 원형의 영역
    - **사이킷런의 eps 파라미터**
  - **최소 데이터 개수(min points)** : 개별 데이터의 입실론 주변 영역에 포함되는 타 데이터의 개수
    - **사이킷런의 min_samples 파라미터**
  - **위 두 개의 파라미터를 통해 최적의 군집을 찾는 게 중요**하다.

- 센트로이드 : 특정 포인트를 중심으로 모인 샘플을 찾는다

📖 가우시안 혼합모델

https://kdeon.tistory.com/35

- 군집화를 적용하고자 하는 하는 방식이다.

  데이터가 여러 개의 가우시안 분포(정규분포)를 가진 데이터 집합들이 섞여서 생성된 것이라는 가정 하에 군집화를 수행
  - **하나의 가우시안 분포에서 생성된 모든 샘플은 하나의 cluster를 형성**한다.
  - 일반적으로 이 **cluster는 타원형**이다.

```python
from sklearn.mixture import GaussianMixture

gm = GaussianMixture(n_components=3, n_init=10, random_state=42)
gm.fit(X)
```

```python
gm.weights_
```

=> array([0.20965228, 0.4000662 , 0.39028152])

- 실제 이 데이터를 생성하기 위해 사용한 가중치는 0.2, 0.4, 0.4

- **K-means 알고리즘의 단점을 보완**해줄 수 있는 모델이다.
  - **GMM**은 k-means보다 **유연하게 다양한 데이터 셋에 잘 적용**될 수 있다. (그러나 **수행 시간이 오래 걸림**)

- **이상치 탐지**에도 사용할 수 있다.
- **사이킷런의 GaussianMixture 클래스**를 사용해서 구현이 가능하다.
  - 단, GMM의 경우 **cluster 개수(k)를 지정해주어야 한다.**
  - **그렇다면 cluster의 개수(k)를 어떻게 지정해주면 좋을까? (즉, GMM의 성능 지표는 무엇일까?)**
  - **→ AIC, BIC와 같은 이론적 정보 기준을 최소화 하는 모델을 선택!!**

<EM(기댓값-최대화) 알고리즘>

- **GMM의 모수를 추정**하기 위해 사용하는 알고리즘이다.
  - 여기서 말하는 **모수 추정**은 대표적으로 **다음의 2가지를 추정**하는 것이다.
    - **개별 정규 분포의 평균과 분산**
    - **각 데이터가 어떤 정규분포에 해당되는지의 확률**