3.1 이론

3.1.1 비지도학습이란?

비지도학습(Unsupervised Learning)은 데이터에 **정답(label)이 존재하지 않는 상황**에서 작동하는 학습 방법입니다.

즉, 데이터의 입력값만 주어지고, 모델은 **스스로 그 안의 패턴이나 구조를 찾아내는 방식**입니다.

지도학습과 비교

구분	지도학습 (Supervised Learning)	비지도학습 (Unsupervised Learning)
입력 데이터	입력값 + 정답(레이블) 존재	입력값만 존재
목적	정답 예측	패턴 발견, 구조 분석
예시	스팸 메일 분류, 가격 예측	고객 세분화, 데이터 시각화

3.1.2 비지도학습의 대표적인 목적

비지도학습은 다음과 같은 두 가지 대표 목적을 갖고 사용됩니다.

1. 군집화 (Clustering)

- 유사한 데이터를 **서로 묶어 그룹화**하는 작업
- 데이터에 숨겨진 **잠재적인 클래스나 유형**을 알아내는 데 효과적

활용 예시:

- 마케팅: 고객 세분화
- 생물학: 유전자 발현 패턴 그룹화
- 추천 시스템: 사용자 행동 기반 그룹 분류

2. 차원 축소 (Dimensionality Reduction)

- 고차원 데이터를 **핵심 정보만 유지하면서 저차원으로 압축**
- 분석 효율과 시각화 용이성 향상

활용 예시:

- 데이터 전처리: 노이즈 제거
- 시각화: 2D/3D로 투영하여 분포 파악
- 학습 효율: 모델 훈련 시간 단축

3.1.3 고차원 공간의 문제점과 차원 축소의 필요성

데이터의 **차원이 많아질수록** (즉, 특성의 개수가 많아질수록) 다음과 같은 문제가 발생합니다. 이를 "차원의 저주(Curse of Dimensionality)"라고 합니다.

차원의 저주(Curse of Dimensionality)

1. **데이터가 희소(Sparse)**: 고차원 공간에서는 데이터가 넓게 퍼져서 유사한 점을 찾기 어려움
2. **거리 측정 불안정**: 유클리디안 거리 등이 신뢰성이 떨어짐
3. **계산 비용 증가**: 차원이 높을수록 학습 시간이 급격히 늘어남
4. **시각화 불가능**: 3차원 이상부터는 시각적으로 이해하기 매우 어려움

→ 따라서 **차원 축소 기법**을 통해 **의미 있는 정보만 유지하면서 차원을 줄이는 것**이 중요합니다.

3.1.4 군집화와 차원 축소의 관계

- **차원 축소 후 군집화**: 차원을 줄인 뒤 군집화를 하면 성능 향상
- **군집화 후 차원 축소**: 군집별로 어떤 특징이 있었는지 시각화 가능

예를 들어, 100차원의 고객 데이터를 먼저 PCA나 t-SNE로 2차원으로 줄이고, 그 위에 K-means 군집을 시각화하면 다음과 같은 이점이 있습니다:

- 사람이 직접 **클러스터 해석** 가능
- **고객 유형별 특성**을 쉽게 파악할 수 있음
- 시각적으로 **군집의 경계와 분포**를 파악 가능

3.1.5 주요 비지도 학습 알고리즘 분류

알고리즘	분류	특징
K-means	군집화	중심 기반 클러스터링
DBSCAN	군집화	밀도 기반, 이상치 탐지
PCA	차원 축소	선형 투영 기반
t-SNE	차원 축소	비선형 시각화 특화

3.2 수식

3.2.1 K-평균 클러스터링 (K-means Clustering)

K-means는 주어진 데이터를 사용자가 정한 K개의 그룹(군집)으로 나누는 비지도학습 알고리즘입니다. 각 군집은 하나의 중심점(centroid)을 가지고 있고, 데이터는 이 중심점들과의 거리를 기준으로 군집에 할당됩니다.

이 알고리즘의 핵심 목표는 군집 내 데이터들이 중심점으로부터 얼마나 가까운지를 나타내는 총 거리 제곱합(SSE)을 최소화하는 것입니다.

비용 함수

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

- x_i: 데이터 포인트
- μ_k: k 번째 클러스터의 중심점
- C_k: k 번째 클러스터에 속한 데이터 집합

이 수식은 각 데이터가 자기 클러스터 중심과 얼마나 떨어져 있는지를 계산하고, 이를 모두 더한 값입니다. 이 값이 작을수록 같은 클러스터 안에 있는 데이터들이 서로 가깝게 모여 있음을 의미합니다.

알고리즘 단계

- K개의 중심점을 무작위로 초기화합니다.
- 각 데이터 포인트 x_i를 가장 가까운 중심점 μ_k에 할당합니다.
- 새롭게 할당된 군집을 기준으로 중심점을 다시 계산합니다.

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

> k 번째 클러스터에 있는 모든 점들의 좌표 평균을 구하는 것입니다.

- 중심점이 더 이상 크게 이동하지 않을 때까지 2–3번 단계를 반복합니다.

특징 및 한계

- 클러스터 수 K를 사용자가 정해야 합니다.
- 클러스터의 **초기 중심값**에 따라 결과가 달라질 수 있습니다.
- 둥근(구형) 형태의 군집에 적합합니다.
- 이상치(outlier)에 매우 민감합니다.

3.2.2 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN은 K-means와 달리 클러스터 수 K를 미리 정하지 않고, 데이터의 밀도(density)에 기반하여 클러스터를 형성합니다. DBSCAN의 핵심 아이디어는, 밀도가 높은 곳은 하나의 군집으로 보고, 밀도가 희박한 지역은 클러스터가 아닌 잡음(noise)으로 처리한다는 것입니다.

주요 개념

- ε: 반경 (이웃으로 간주할 최대 거리)
- minPts: 해당 반경 안에 있어야 할 최소 점 개수

DBSCAN은 각 포인트를 다음 세 가지 중 하나로 분류합니다:

1. **Core Point**: 반경 ε 안에 minPts 이상의 점이 존재하는 경우

2. **Border Point**: Core Point에 인접하지만, 자기 주변에는 minPts 미만
3. **Noise Point**: 어떤 Core Point에도 속하지 않는 이상치

밀도 기반 접근

어떤 점 x의 반경 ε 안에 있는 점들의 집합은 다음과 같이 정의됩니다.

$$N_\varepsilon(x) = \{y \in \mathbb{R}^d \mid ||x - y|| \leq \varepsilon\}$$

> d 차원 공간 위에서, 점 x 로 부터 ε 이하 거리에 있는 모든 점 y 들을 모은 집합

- $||x - y||$: 두 점 사이의 거리 (보통 유클리디안 거리)
- 이 집합 $N_\varepsilon(x)$ 에 속한 점의 수가 **minPts 이상**이면 x는 Core Point

이웃 점의 개수가 minPts 이상이면, 해당 점은 Core Point로 간주하고 군집 확장을 시작합니다. DBSCAN은 이러한 점들을 연결해 가며 밀도가 높은 영역을 하나의 클러스터로 만들어냅니다.

장단점

- 클러스터의 모양이 **복잡하거나 불규칙한 경우**에도 잘 작동
- 이상치(Noise)를 따로 분리해낼 수 있음
- K값을 정할 필요가 없음
- 하지만 ε 와 minPts를 잘못 설정하면 성능이 크게 저하될 수 있음

3.2.3 PCA (주성분 분석, Principal Component Analysis)

PCA는 고차원 데이터를 핵심 정보만 유지하면서 **저차원으로 변환**하는 차원 축소 기법입니다. 데이터의 분산(variability)을 최대한 보존하는 새로운 좌표축을 찾아 데이터를 이 축들 위로 재표현합니다.

과정

데이터 중심화: 평균을 0으로 맞추기 위해 각 데이터에서 평균을 뺍니다. 이 과정을 통해 데이터는 원점을 중심으로 재배열됩니다.

$$x_i' = x_i - \bar{x}$$

$$x_i = \begin{bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ \vdots \\ x_i^{(d)} \end{bmatrix} \in \mathbb{R}^{d \times 1}$$

$$x_i' = \begin{bmatrix} x_i^{(1)} - \bar{x}^{(1)} \\ x_i^{(2)} - \bar{x}^{(2)} \\ x_i^{(3)} - \bar{x}^{(3)} \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

공분산 행렬 계산:

$$\Sigma = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i' (\boldsymbol{x}_i')^T \in \mathbb{R}^{d \times d}$$

$$x_i'(x_i')^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac \\ ba & b^2 & bc \\ ca & cb & c^2 \end{bmatrix}$$

$$x_i'(x_i')^T = \begin{bmatrix} (x_i^{(1)} - \bar{x}^{(1)})^2 & (x_i^{(1)} - \bar{x}^{(1)})(x_i^{(2)} - \bar{x}^{(2)}) & (x_i^{(1)} - \bar{x}^{(1)})(x_i^{(3)} - \bar{x}^{(3)}) \\ (x_i^{(2)} - \bar{x}^{(2)})(x_i^{(1)} - \bar{x}^{(1)}) & (x_i^{(2)} - \bar{x}^{(2)})^2 & (x_i^{(2)} - \bar{x}^{(2)})(x_i^{(3)} - \bar{x}^{(3)}) \\ (x_i^{(3)} - \bar{x}^{(3)})(x_i^{(1)} - \bar{x}^{(1)}) & (x_i^{(3)} - \bar{x}^{(3)})(x_i^{(2)} - \bar{x}^{(2)}) & (x_i^{(3)} - \bar{x}^{(3)})^2 \end{bmatrix}$$

- 중심화된 각 벡터 x_i'에 대해 자기 자신과의 외적(Outer Product)을 계산한 후 평균을 냅니다.

- 결과는 $d \times d$ 크기의 공분산 행렬로, 각 성분은 특성 간의 상관 관계를 나타냅니다.

고유값 분해를 통해 분산이 큰 방향(고유벡터)을 찾습니다.

$$\Sigma = Q \Lambda Q^T$$

- Σ를 고유값 분해하면, Q는 고유벡터(Eigenvectors)를 열벡터로 가지는 직교 행렬,
  Λ는 고유값(Eigenvalues)이 대각선에 위치한 대각 행렬입니다.
- 고유값이 클수록 해당 고유벡터 방향으로의 분산이 큽니다.

고유값이 큰 순서대로 k개를 선택

- 고유값을 내림차순으로 정렬하고, **상위 k개의 고유벡터**를 선택합니다.
- 이 고유벡터들은 데이터가 가장 많이 퍼진 방향을 의미하며, 새로운 축(주성분)이 됩니다.
- 선택된 k개의 고유벡터로 구성된 행렬을 $W \in \mathbb{R}^{d \times k}$라고 하면:

데이터 투영

$$z_i = W^T x_i'$$

- 중심화된 원래 데이터 를 주성분 축 W 위에 투영하여 새로운 표현 z_i를 얻습니다.
- 결과는 k-차원 데이터로 변환된 것입니다.

해석

- 고유값이 클수록 그 방향에 정보가 많이 담겨 있다는 뜻
- 차원을 줄여도 정보 손실이 적습니다 (적절한 k 선택 시)
- PCA는 **선형 구조를 가정**하며, 전체적인 데이터 분포를 설명하려고 합니다

단계	설명	수식 (미리보기)	차원
1	데이터 중심화 (평균 0으로 이동)	$x_i' = x_i - \bar{x}$	$\bar{x} \in \mathbb{R}^d, \quad x_i' \in \mathbb{R}^d$
2	공분산 행렬 계산	$\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i' (x_i')^T$	$\Sigma \in \mathbb{R}^{d \times d}$
3	고유값 분해	$\Sigma = Q \Lambda Q^T$	$Q \in \mathbb{R}^{d \times d}, \quad \Lambda \in \mathbb{R}^{d \times d}$
4	주성분 선택 (상위 k개)	$W = [v_1, v_2, \ldots, v_k]$	$W \in \mathbb{R}^{d \times k}$
5	데이터 투영 (저차원 표현)	$z_i = W^T x_i'$	$z_i \in \mathbb{R}^k$

3.2.4 t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE는 주로 **고차원 데이터를 2D나 3D로 시각화**하기 위한 알고리즘입니다.

PCA와는 달리, 데이터 간의 **전체 구조**보다는 국소 구조(local structure)를 잘 보존하도록 설계되어 있습니다.

즉, **서로 가까운 점은 시각화 결과에서도 가깝게**, 먼 점은 되도록 멀리 유지하는 것이 목표입니다. 고차원 데이터의 구조와 패턴을 유지하면서, 차원 축소를 가능하게 합니다.

기본 아이디어

1. 고차원 공간에서 각 점 x_i와 x_j 사이의 거리로 유사도 확률 p_{ij} 계산
2. 저차원 공간에서는 y_i, y_j로 대응하고, 비슷한 방식으로 q_{ij} 계산
3. 두 확률 분포 사이의 차이(KL Divergence)를 최소화

$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

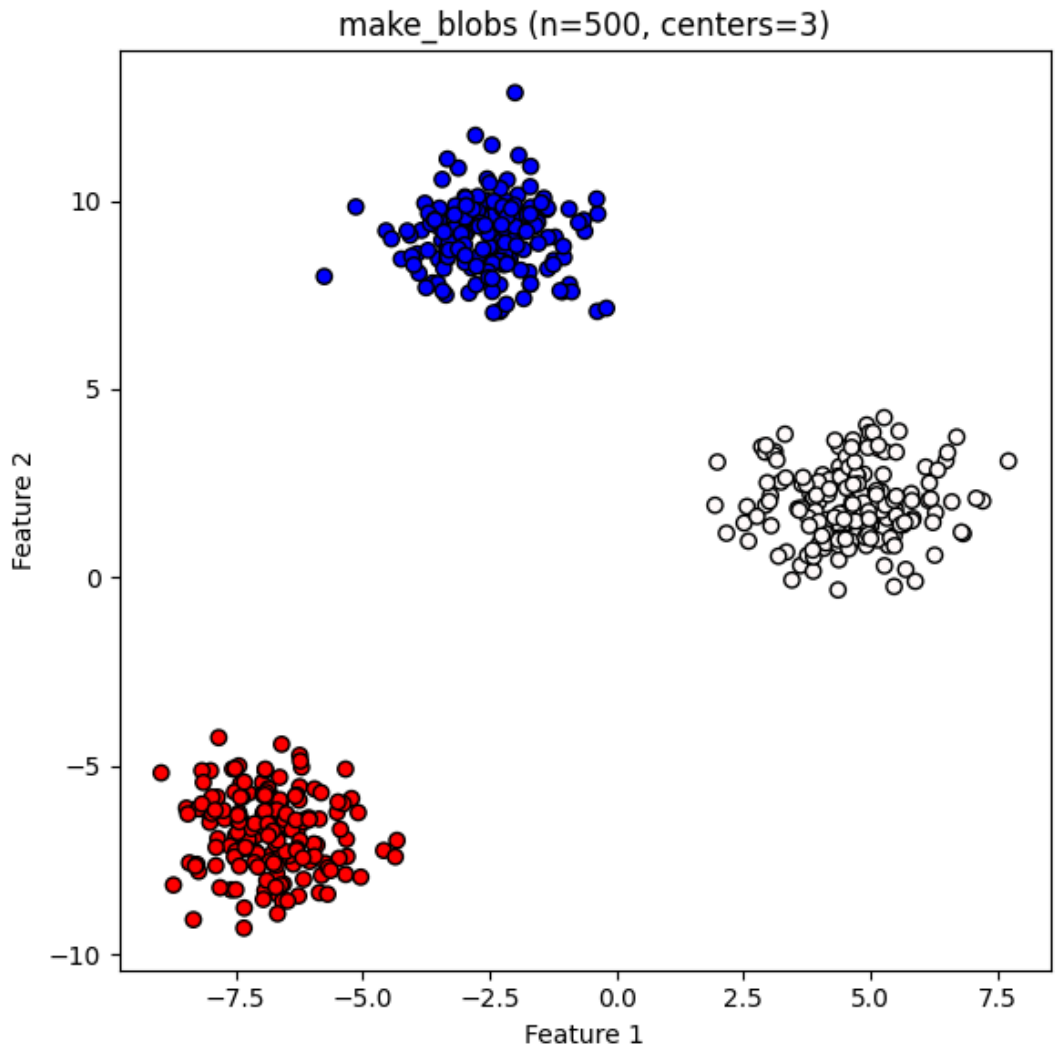
저차원 공간에서는 t-분포를 사용하여, 데이터가 너무 밀집되지 않도록 합니다.

특징

- **비선형 차원 축소 기법**이며, 복잡한 데이터 구조 시각화에 탁월
- 데이터가 시각적으로 뚜렷한 클러스터 구조를 띠도록 만들어줌
- 학습 시 랜덤 요소가 있어서 결과가 매번 조금씩 달라질 수 있음
- **속도 느림**, 특히 대용량 데이터에서는 최적화 필요

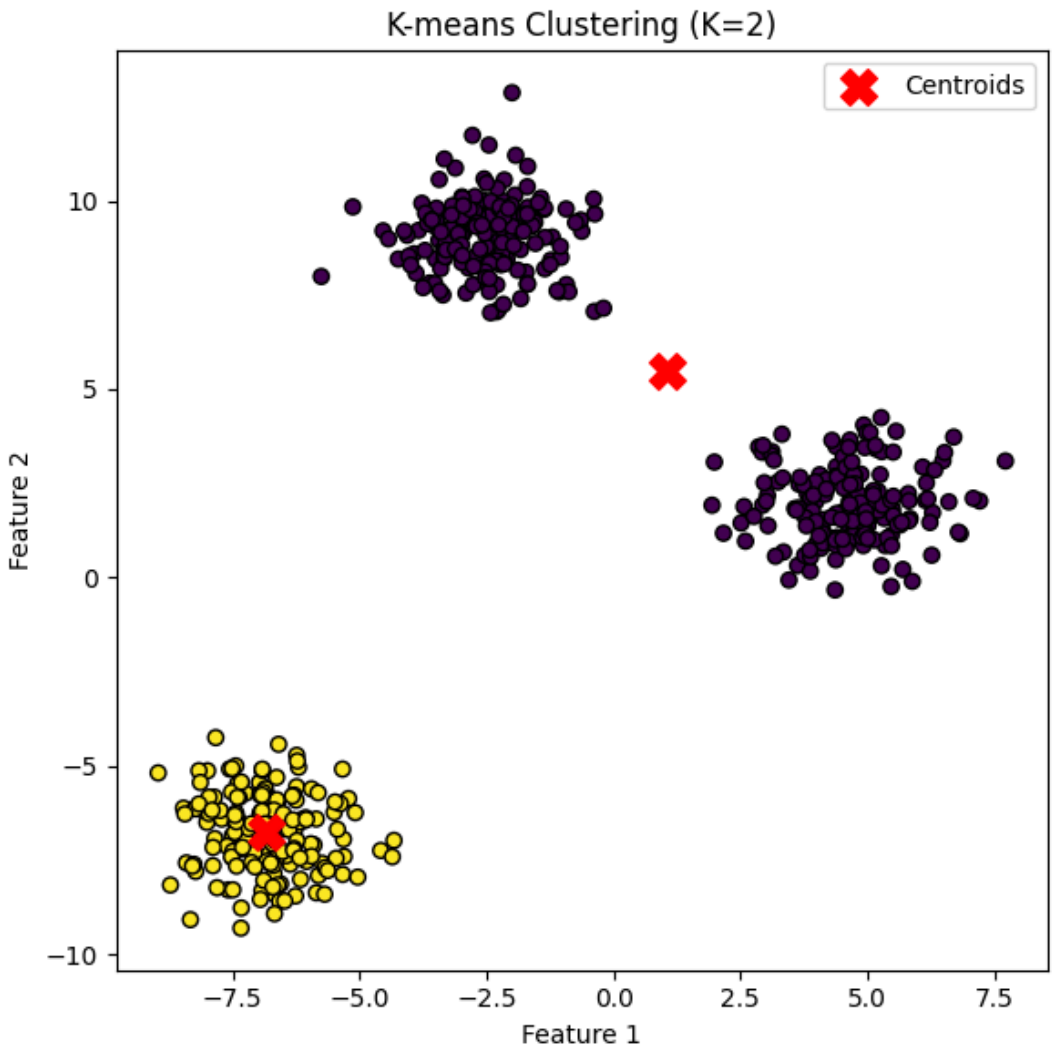
3.3 코드

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# 데이터: make_blobs (명확한 군집 구조)
X, y = make_blobs(n_samples=500, centers=3, cluster_std=1.0, random_state=42)

# 시각화
plt.figure(figsize=(6, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='bwr', edgecolor='k')
plt.title('make_blobs (n=500, centers=3)')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.tight_layout()
plt.show()
```




```
from sklearn.cluster import KMeans

# K-means 클러스터링 (클러스터 수 K=2)
kmeans = KMeans(n_clusters=2, random_state=42)
y_kmeans = kmeans.fit_predict(X)

# 시각화
plt.figure(figsize=(6, 6))
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, cmap='viridis', edgecolor='k')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            s=200, c='red', marker='X', label='Centroids')
plt.title('K-means Clustering (K=2)')
plt.xlabel('Feature 1')
```

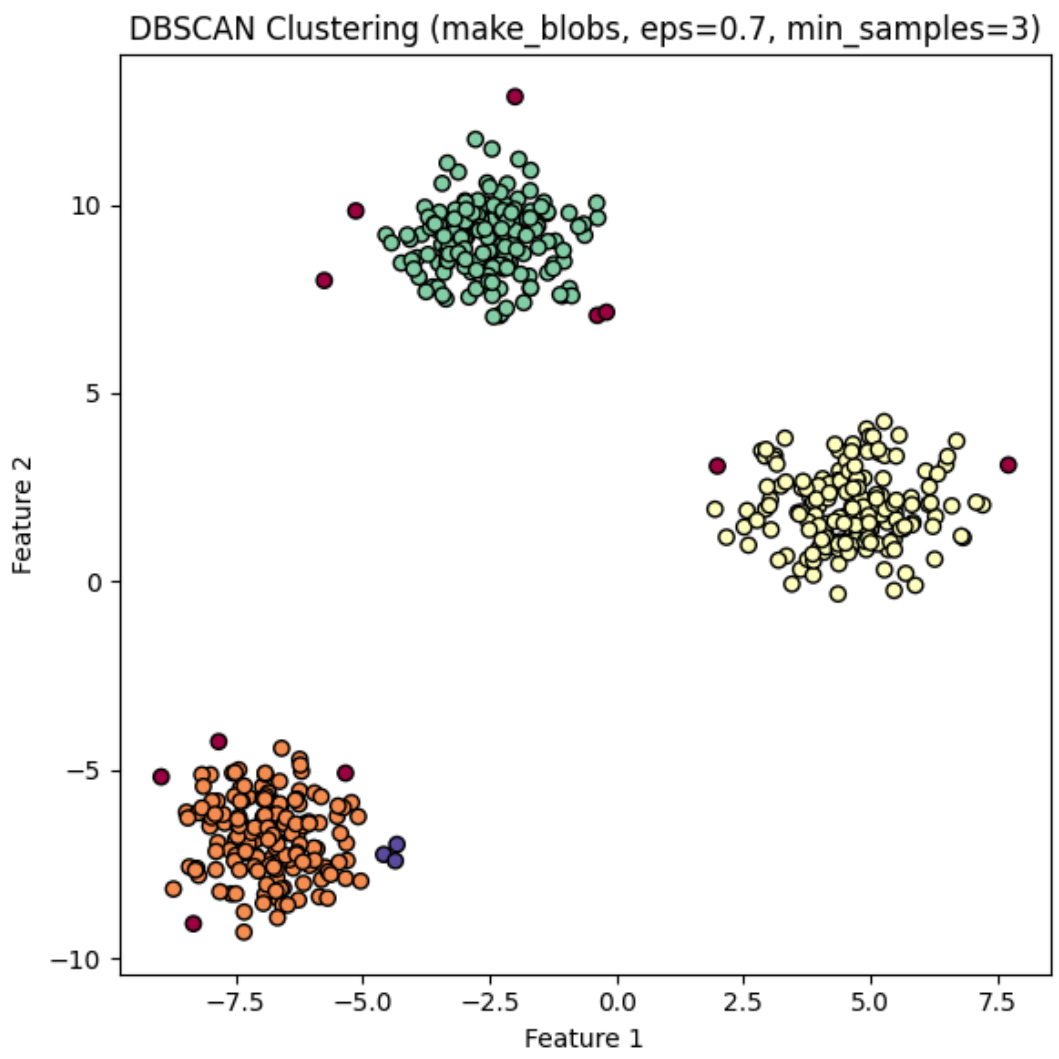
```
plt.ylabel('Feature 2')
plt.legend()
plt.tight_layout()
plt.show()
```



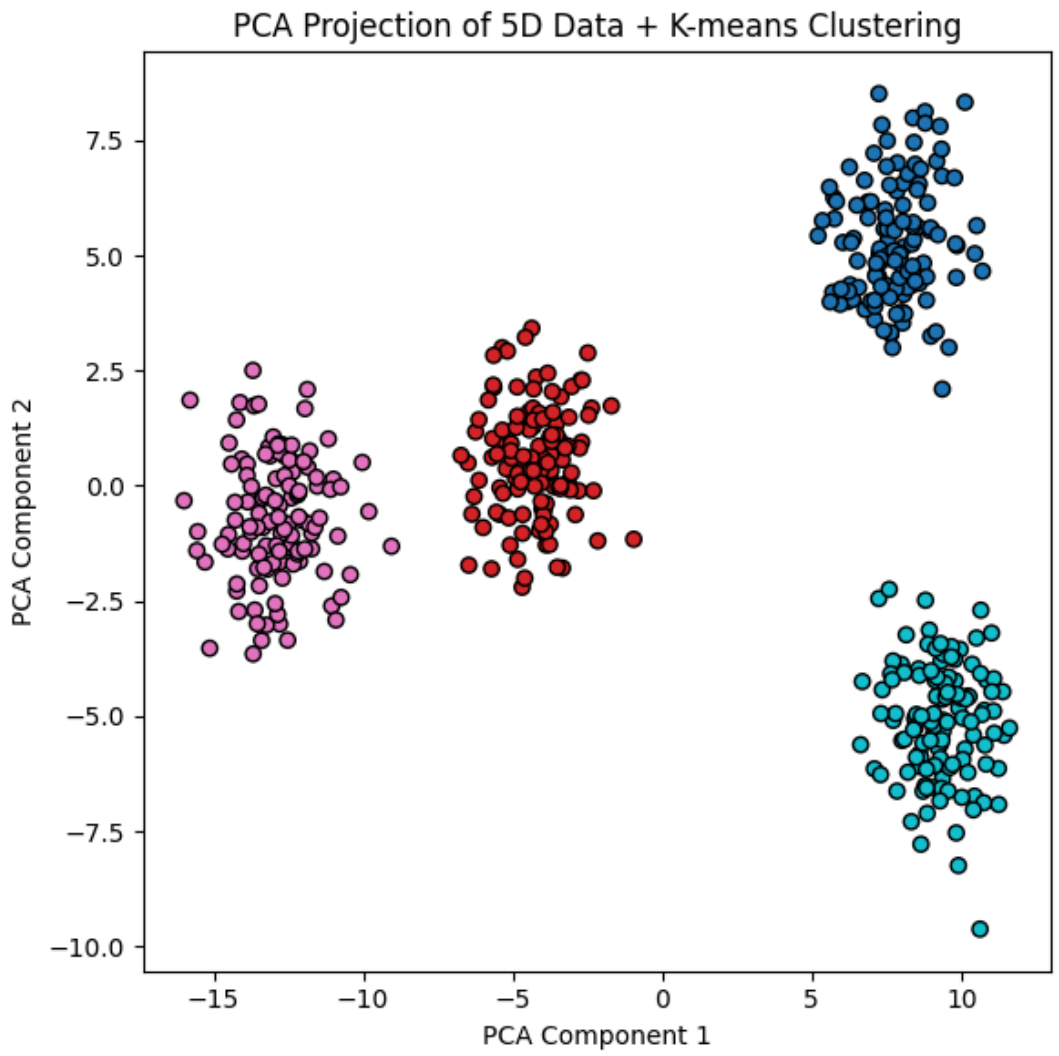

```
from sklearn.metrics import silhouette_score, adjusted_rand_score

# 실루엣 점수 계산 (클러스터링 품질 평가 지표, 1에 가까울수록 좋음)
silhouette = silhouette_score(X, y_kmeans)


print(f"K-means 성능 평가:")
print(f" - 실루엣 점수 (Silhouette Score): {silhouette:.3f}")
'''
군집 내 응집도와 군집 간 분리도를 함께 고려하는 지표로, 1에 가까울수록 좋습니다.
'''
```

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.7, min_samples=3)
y_dbscan = dbscan.fit_predict(X)

# 시각화
plt.figure(figsize=(6, 6))
plt.scatter(X[:, 0], X[:, 1], c=y_dbscan, cmap='Spectral', edgecolor='k')
plt.title('DBSCAN Clustering (make_blobs, eps=0.7, min_samples=3)')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.tight_layout()
plt.show()

# 실루엣 점수 계산 (Noise 제외)
```

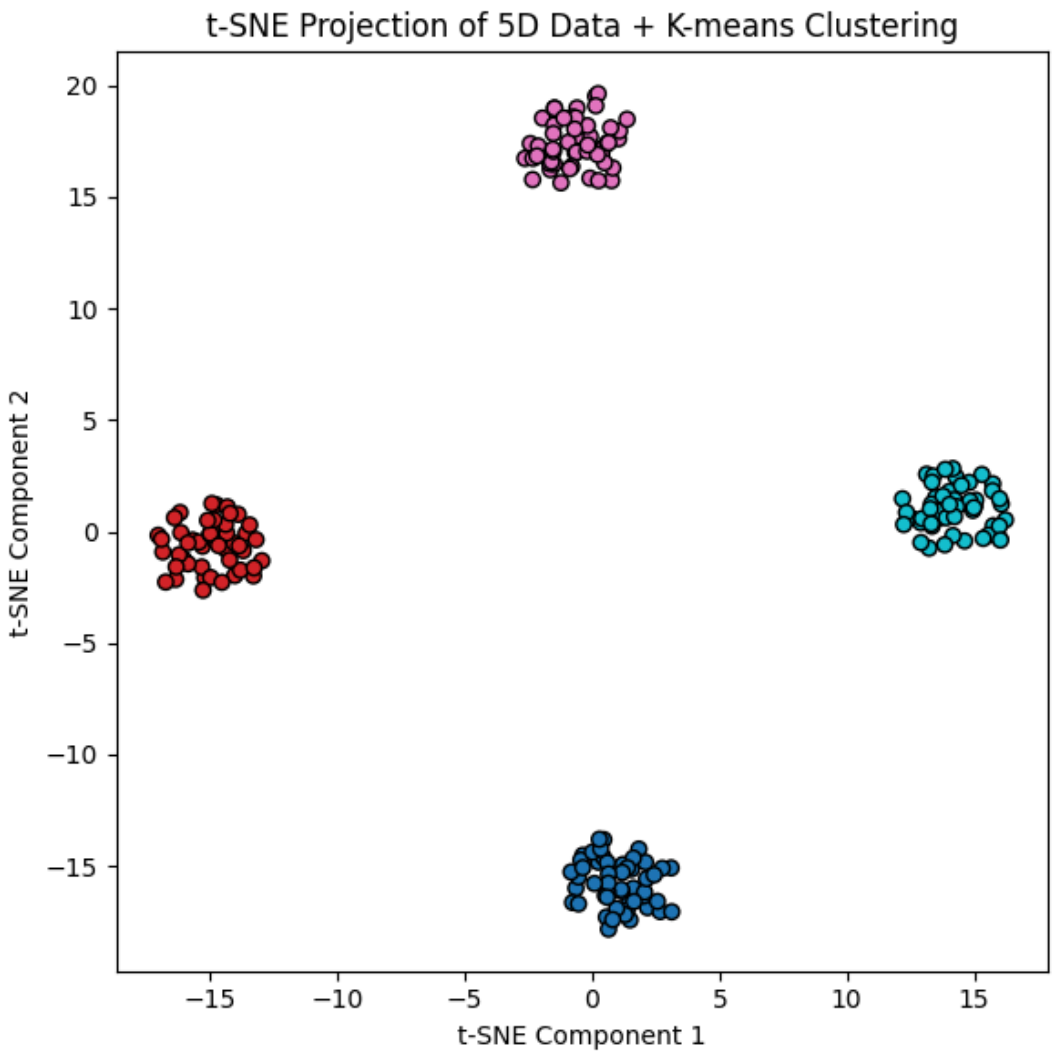
```
mask = y_dbscan != -1
if len(set(y_dbscan[mask])) > 1:
    silhouette = silhouette_score(X[mask], y_dbscan[mask])
else:
    silhouette = float('nan')

print(f"DBSCAN 실루엣 점수 (eps=0.7, min_samples=3): {silhouette:.3f}")
```




```
from sklearn.decomposition import PCA

# 5차원 고차원 데이터 생성
X_highdim, y_highdim = make_blobs(n_samples=500, centers=4, n_features=5, cluster_std=1.2, random_state=42)

# PCA로 2차원 축소
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_highdim)

# K-means 클러스터링 (고차원 공간에서 수행)
kmeans = KMeans(n_clusters=4, random_state=42)
y_kmeans = kmeans.fit_predict(X_highdim)

# 시각화
plt.figure(figsize=(6, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_kmeans, cmap='tab10', edgecolor='k')
plt.title('PCA Projection of 5D Data + K-means Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.tight_layout()
plt.show()
```




```
from sklearn.datasets import make_blobs
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# 5차원 데이터
X_highdim, y_highdim = make_blobs(n_samples=200, centers=4, n_features=5, cluster_std=1.2, random_state=42)

# K-means 군집
kmeans = KMeans(n_clusters=4, random_state=42)
y_kmeans = kmeans.fit_predict(X_highdim)

# t-SNE 임베딩
tsne = TSNE(n_components=2, perplexity=30, n_iter=500, random_state=42)
X_tsne = tsne.fit_transform(X_highdim)

# 시각화
plt.figure(figsize=(6, 6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_kmeans, cmap='tab10', edgecolor='k')
plt.title('t-SNE Projection of 5D Data + K-means Clustering')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.tight_layout()
plt.show()
```


t-SNE Projection of 5D Data + K-means Clustering
t-SNE Component 2
20
15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
t-SNE Component 1

3.4 결과 해석

3.4.1 k-means 군집 평가 및 시각화

오차제곱합(Sum of Squared Errors, SSE)

- 각 클러스터 내 샘플이 해당 클러스터 중심으로부터 얼마나 흩어져 있는지 수치화한 지표.
- 수식

$$SSE = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

구현

- `sklearn.cluster.KMeans`의 `inertia_` 속성 사용
- 여러 K에 대해 SSE를 계산한 뒤, K값에 따른 SSE 변화를 엘보(elbow) 그래프로 표시

실루엣 계수(Silhouette Coefficient)

- 각 샘플의 응집도(a)와 분리도(b)를 기반으로, "군집 내 응집력 대비 다른 군집과의 분리도"를 –1에서 1 사이로 정규화한 점수
- 한 샘플 i에 대한 계산

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

- $a(i)$: 같은 클러스터 내 다른 샘플들과의 평균 거리
- $b(i)$: 가장 가까운 다른 군집의 모든 샘플들과의 평균 거리
- 전체 군집화 품질: 모든 $s(i)$의 평균
- 구현 및 시각화
  - `sklearn.metrics.silhouette_score`로 전체 점수 계산
  - `sklearn.metrics.silhouette_samples`와 matplotlib으로 각 클러스터별 실루엣 플롯 작성

2차원 산점도 시각화

- 원본 특성 공간이 고차원이면, PCA나 t-SNE를 적용해 2차원으로 축소
- 축소된 좌표에 클러스터 레이블별 색상 및 중심점 표시

3.4.2 DBSCAN 군집 평가 및 시각화

핵심 지표

- 추정된 군집 개수 (excluding noise)
- 노이즈 비율: 라벨이 –1로 판정된 샘플의 비중(군집에 포함되지 않음)
- 실루엣 계수: 노이즈 샘플 제외 후 계산

파라미터 민감도 분석

- ε(eps)와 최소 샘플(min_samples) 조합별로 군집 수 및 노이즈 비율 변화
- 파라미터 그리드(grid) 탐색 결과를 Heatmap으로 시각화(eps, min_samples를 변경해 가면서 시도함)

군집 분포 시각화

- 2차원 축소(PCA/t-SNE) 후,
  - Core 샘플: ●
  - Border 샘플: ○
  - Noise 샘플: ×
- 클러스터별 색상을 달리해 표시

3.4.3 PCA 결과 해석 및 시각화

설명 가능한 분산 비율(Explained Variance Ratio)

- 각 주성분이 전체 데이터 분산을 얼마나 설명하는지 측정
- j번째 성분

$$\mathrm{EVR}_j = \frac{\lambda_j}{\sum_{l=1}^{L} \lambda_l}$$

- 스크리(scree) 플롯: 주성분 번호에 따른 EVR 및 누적 EVR 표시

주성분 산점도

- 1차원/2차원 주성분 축에 원본 데이터 투영
- 군집 라벨별 색상으로 분포 확인

피처 로딩스(Loading Scores)

- 각 원본 피처가 주성분에 기여하는 정도
- 주요 피처 상위 5~10개를 막대그래프로 시각화

3.4.4 추가 해석 및 시각화 기법

- t-SNE 등의 비선형 차원 축소 기법으로 임베딩 후 군집 결과 색상 표시
- 병렬 좌표(parallel coordinates) 플롯: 클러스터별 특성 비교
- 클러스터×피처 히트맵: 클러스터 중심의 특성값 시각화