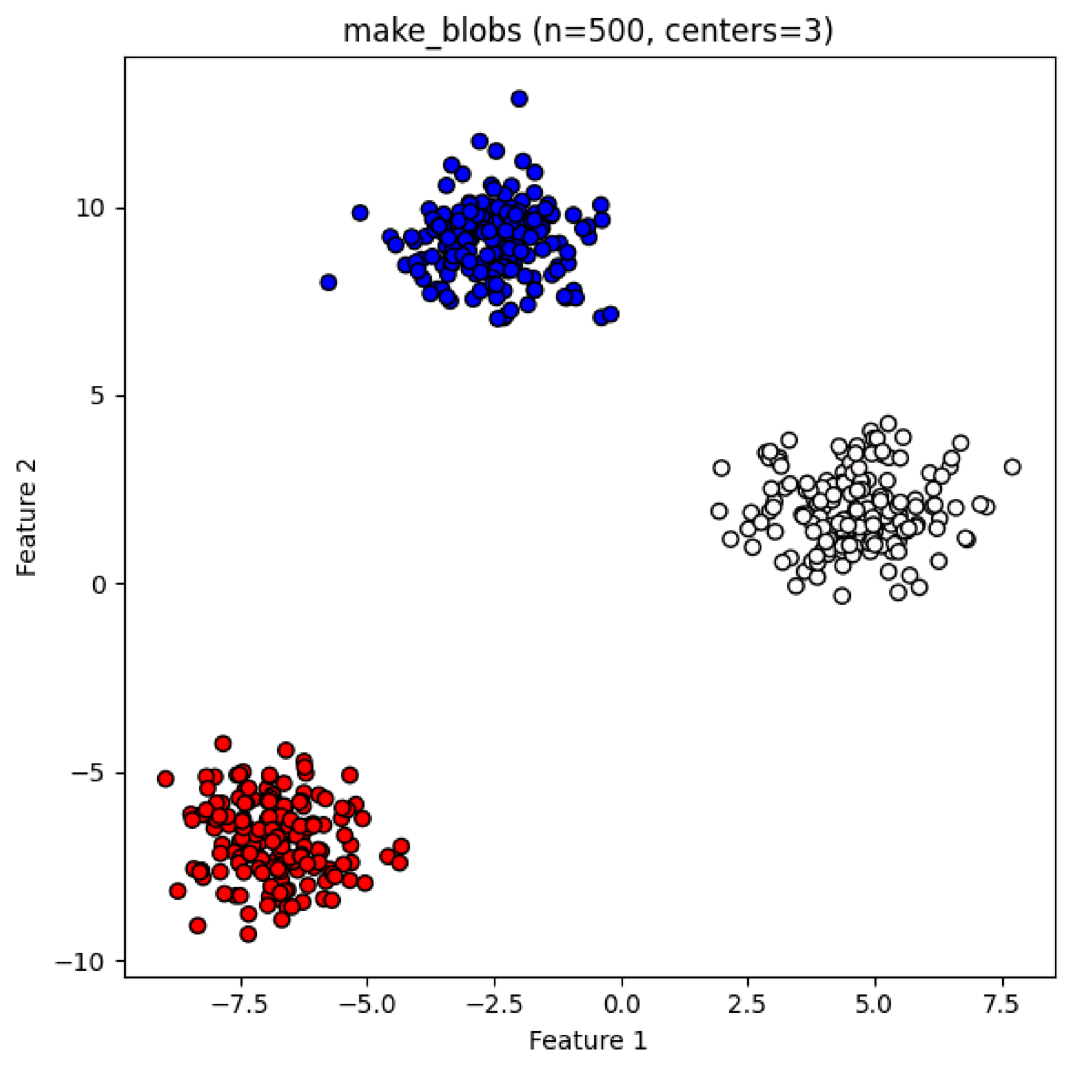
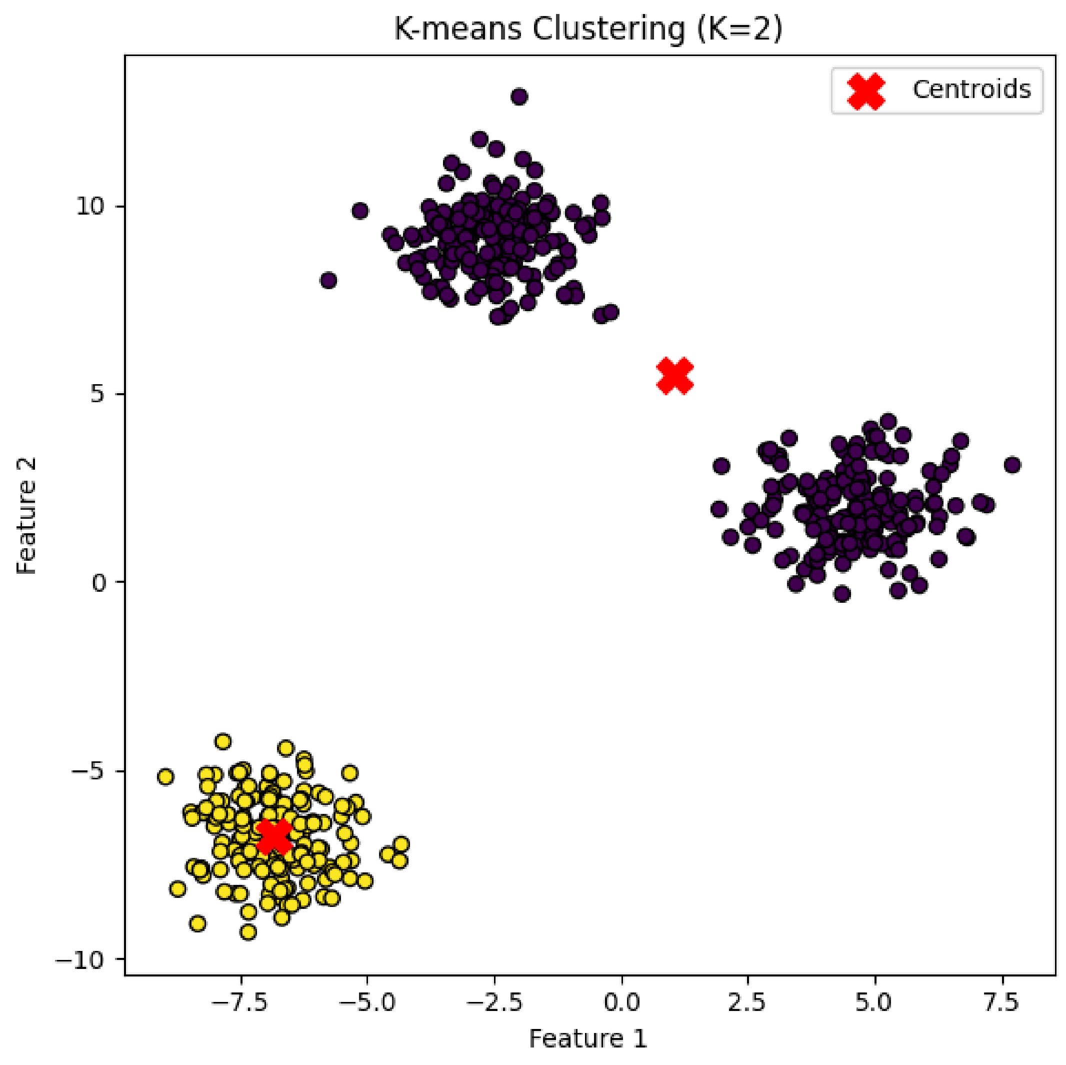
3.3 코드

```
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()
```



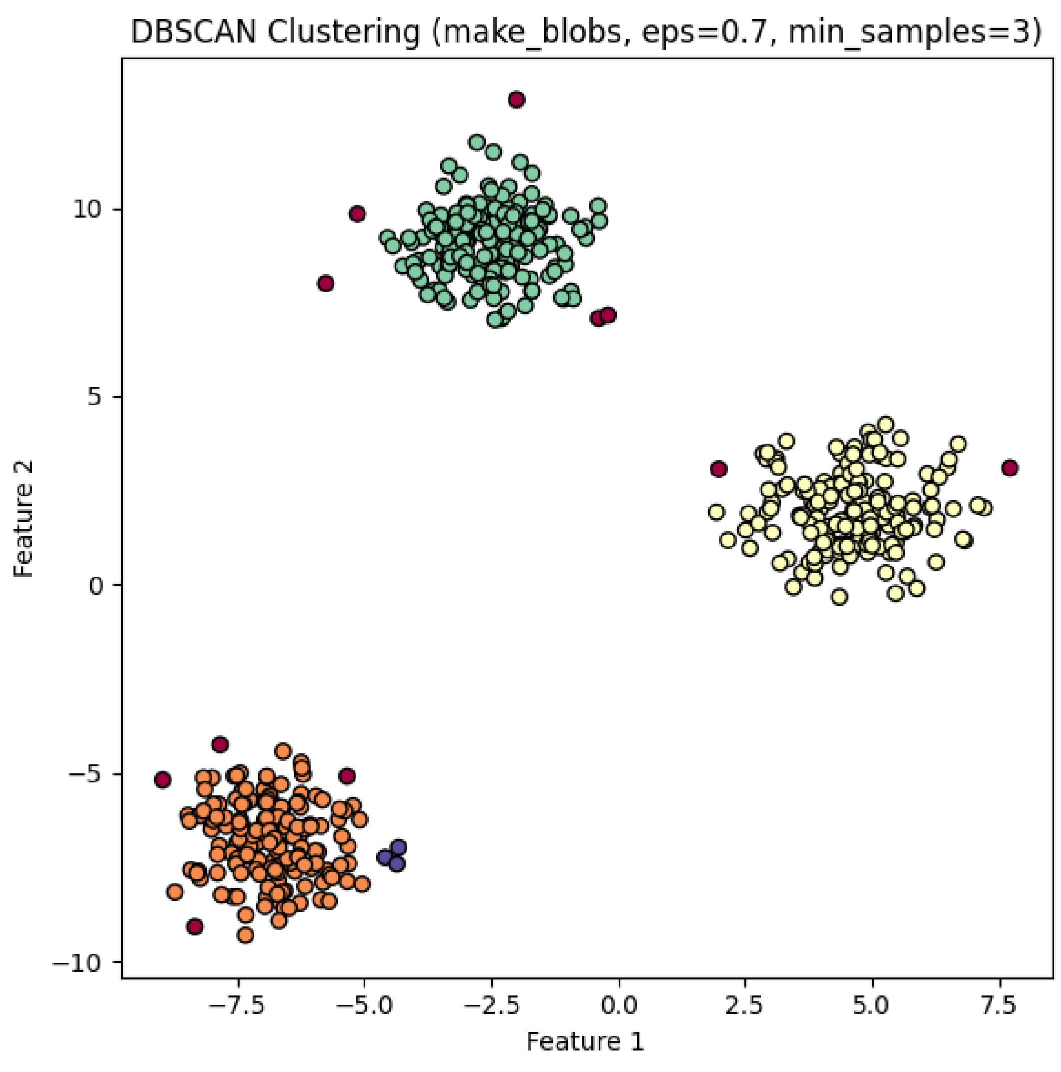
```
from sklearn.cluster import KMeans

# 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()
```



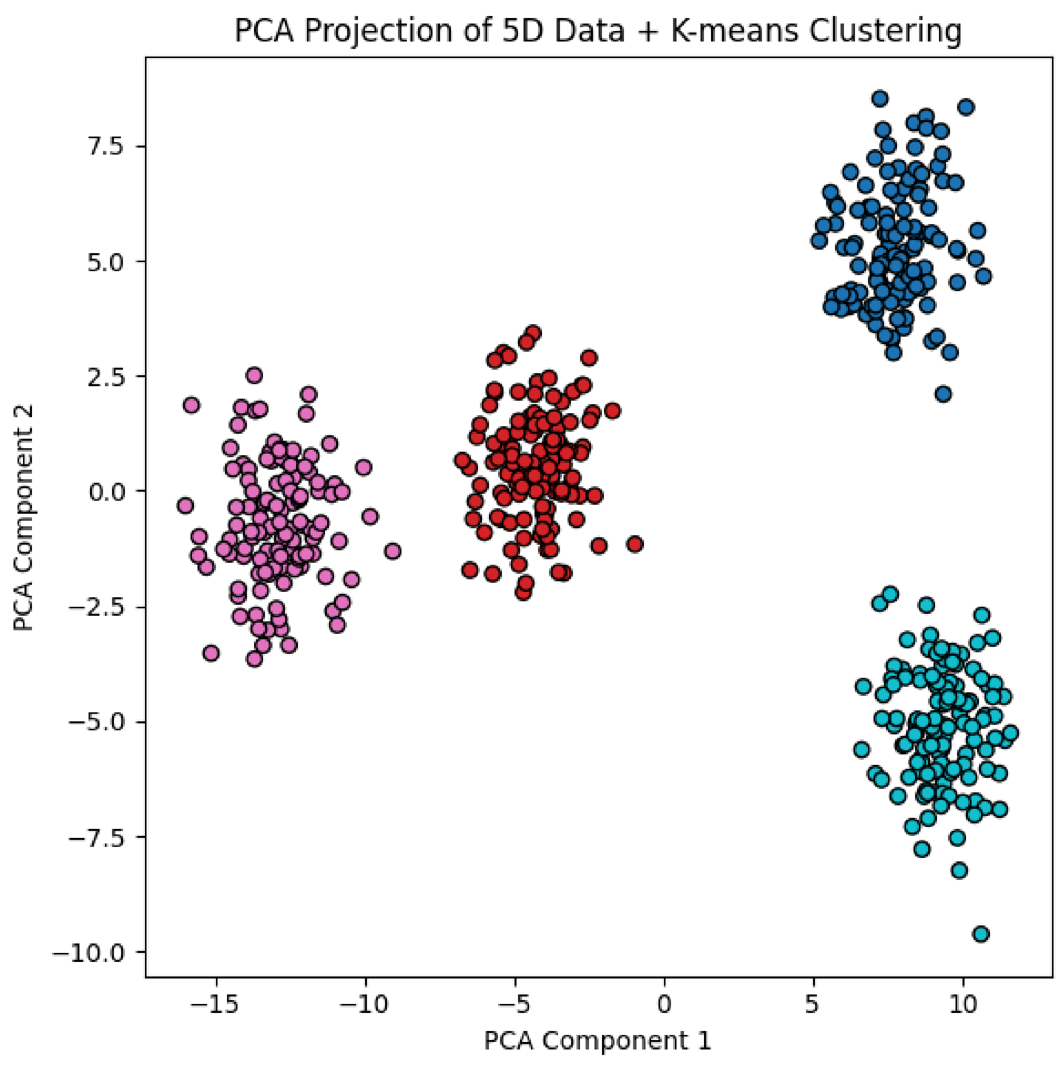
```
from sklearn.metrics import silhouette_score, adjusted_rand_score

# 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)
```

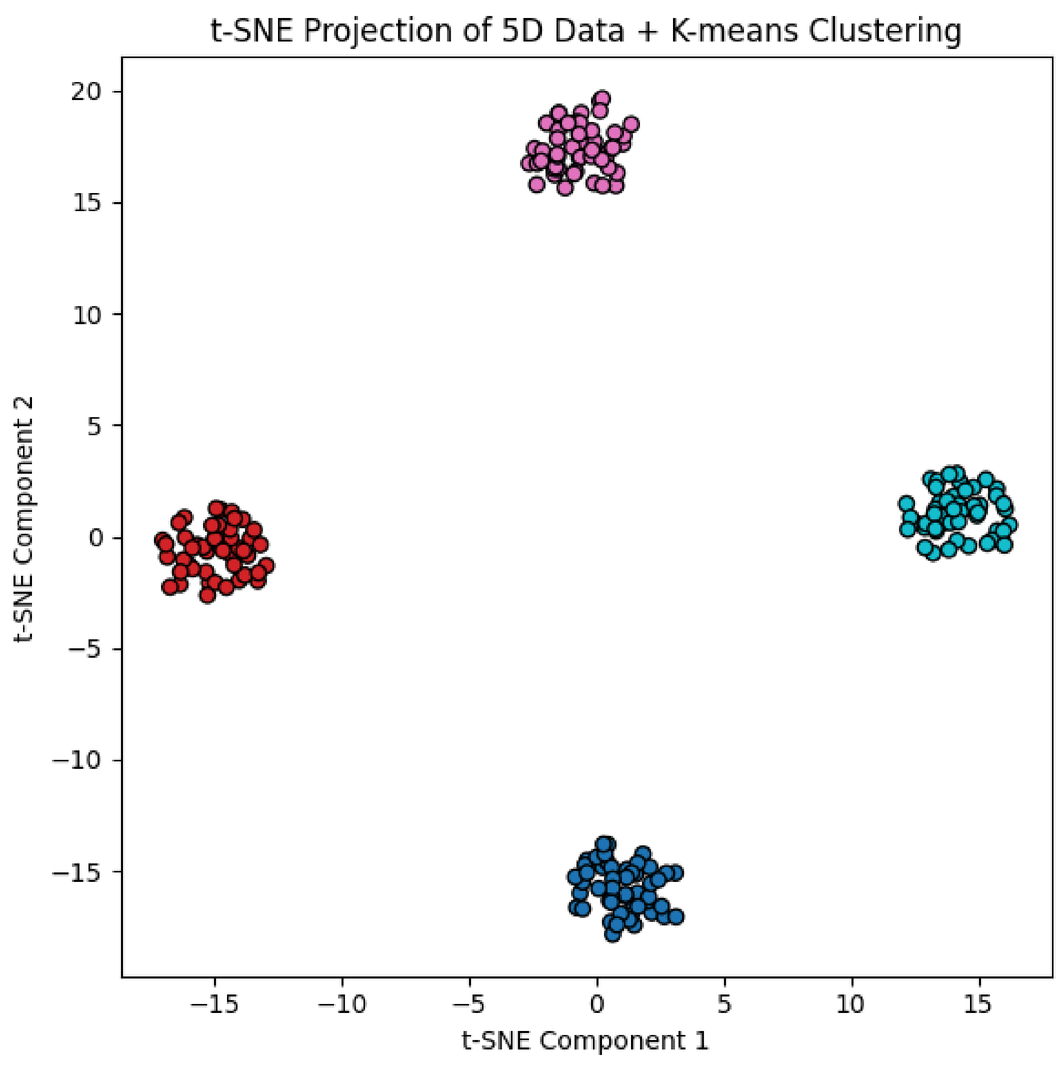
```
mask = y_dbscan != -1
if len(set(y_dbscan[mask])) > 1:
    silhouette = silhouette_score(X[mask], y_dbscan[mask])
else:
    silhouette = float('nan')

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 1
t-SNE Component 2