군집 분석 (Clustering)

- 대표적인 비지도 학습(Unsupervised Learning)의 한 종류로, 정답(label)이 없는 데이터들을 비슷한 특성을 가진 데이터들끼리 그룹(군집, cluster)으로 묶는 과정
- 데이터의 숨겨진 구조나 패턴을 발견하는 것을 목표로 하며, 고객 세분화, 이상 탐지, 이미지 분할 등 다양한 분야에서 활용

주요 군집 분석 알고리즘 종류

1. **K-평균 군집 (K-Means Clustering)**:
    - 데이터를 미리 정해진 K개의 군집으로 나누는 **분할 군집(Partitional Clustering)** 알고리즘
    - 각 군집은 중심점(centroid)을 가지며, 모든 데이터는 가장 가까운 중심점에 속함
    - 알고리즘은 각 군집 내 데이터들과 중심점 간의 거리 제곱의 합(Inertia)을 최소화하는 방향으로 중심점을 반복적으로 업데이트하며 최적의 군집을 탐색
    - **장점**: 간단하고 계산 속도가 빨라 대용량 데이터에도 적용 가능
    - **단점**:
        - 군집의 개수(K)를 사전에 지정 필요
        - 초기 중심점 위치에 따라 결과가 달라질 수 있음
        - 군집이 원형(spherical)이 아닐 경우 잘 작동하지 않으며, 이상치에 민감
2. **계층적 군집 (Hierarchical Clustering)**:
    - 데이터 간의 거리를 기반으로, 가까운 데이터부터 순차적으로 묶어나가며 나무 형태의 계층 구조(덴드로그램, Dendrogram)를 만드는 알고리즘
    - **방식**:
        - **병합적(Agglomerative)**: 각 데이터를 하나의 군집으로 시작하여, 가장 가까운 두 군집을 반복적으로 합쳐나가 최종적으로 하나의 군집이 될 때까지 진행 (일반적으로 사용)
        - **분할적(Divisive)**: 전체 데이터를 하나의 군집으로 시작하여, 가장 의미 없는 군집을 반복적으로 분할
    - **장점**:
        - 군집의 개수 K를 미리 정할 필요가 없음
        - 덴드로그램을 통해 데이터의 구조를 시각적으로 파악하고 원하는 수준에서 군집을 분할 가능
    - **단점**: 모든 데이터 쌍 간의 거리를 계산해야 하므로 계산량이 많아 대용량 데이터에는 부적합
3. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**:
    - 데이터가 밀집된 정도, 즉 **밀도(Density)**를 기반으로 군집을 형성하는 알고리즘
    - K-Means와 달리 복잡한 모양(e.g. 초승달, 도넛)의 군집도 잘 찾아내며, 어떤 군집에도 속하지 않는 데이터를 **노이즈(Noise)** 또는 이상치로 구분 가능
    - **주요 파라미터**:
        - `eps` (epsilon): 하나의 데이터 포인트를 중심으로 하는 이웃의 반경.
        - `min_samples`: 하나의 군집을 형성하기 위해 `eps` 반경 내에 포함되어야 하는 최소 데이터의 개수.
    - **장점**:
        - 복잡한 형태의 군집을 발견 가능

        - 노이즈를 자동으로 처리
        - 군집의 개수를 미리 정할 필요가 없음
    - **단점**:
        - 데이터의 밀도가 다양한 경우, 모든 군집에 적용할 수 있는 최적의 `eps`와 `min_samples`를 찾기 어려움
        - 고차원 데이터에서는 거리 계산의 어려움으로 성능이 저하 가능

코드 예시

주의사항

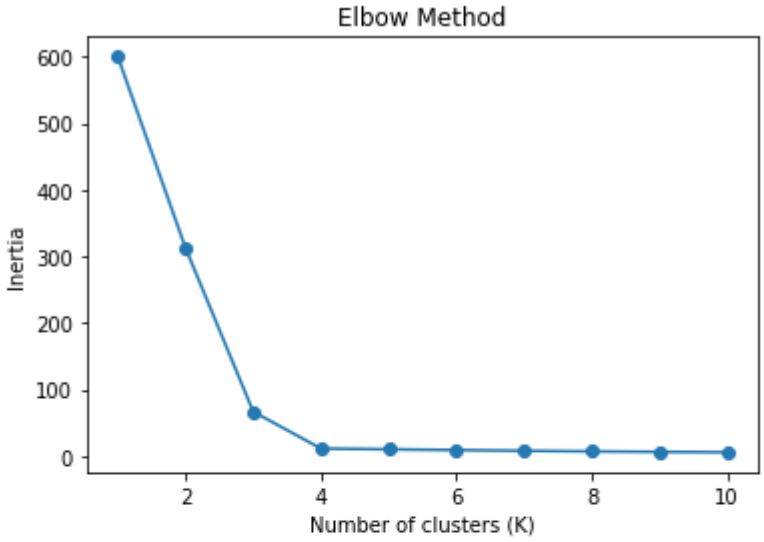
- **특성 스케일링 필수**: 세 알고리즘 모두 거리 기반으로 동작하므로, **반드시 `StandardScaler` 등으로 스케일링**하여 특성들의 단위를 맞춰주어야 합니다.

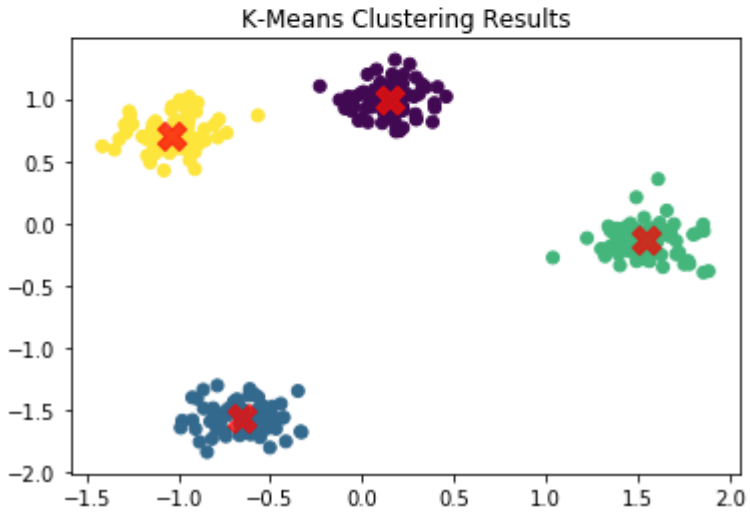
1. K-평균 군집 (K-Means)

- **최적의 K 찾기 (엘보우 방법, Elbow Method)**: K값을 1부터 점차 늘려가면서 각 K에 대한 군집 내 거리 제곱합(Inertia)을 계산하고, 그래프가 팔꿈치처럼 급격히 꺾이는 지점을 최적의 K로 선택합니다.

```
import numpy as np
import pandas as pd
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# 1. 데이터 생성 및 스케일링
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.8, random_state=42)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. 최적의 K 찾기 (엘보우 방법)
inertia_list = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia_list.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia_list, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters (K)')
plt.ylabel('Inertia')
plt.show() # 그래프에서 K=4가 최적임을 확인

# 3. K-Means 모델 학습 및 예측
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(X_scaled)

# 4. 결과 시각화
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=cluster_labels, cmap='viridis',
```

```
marker='o')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title('K-Means Clustering Results')
plt.show()
```



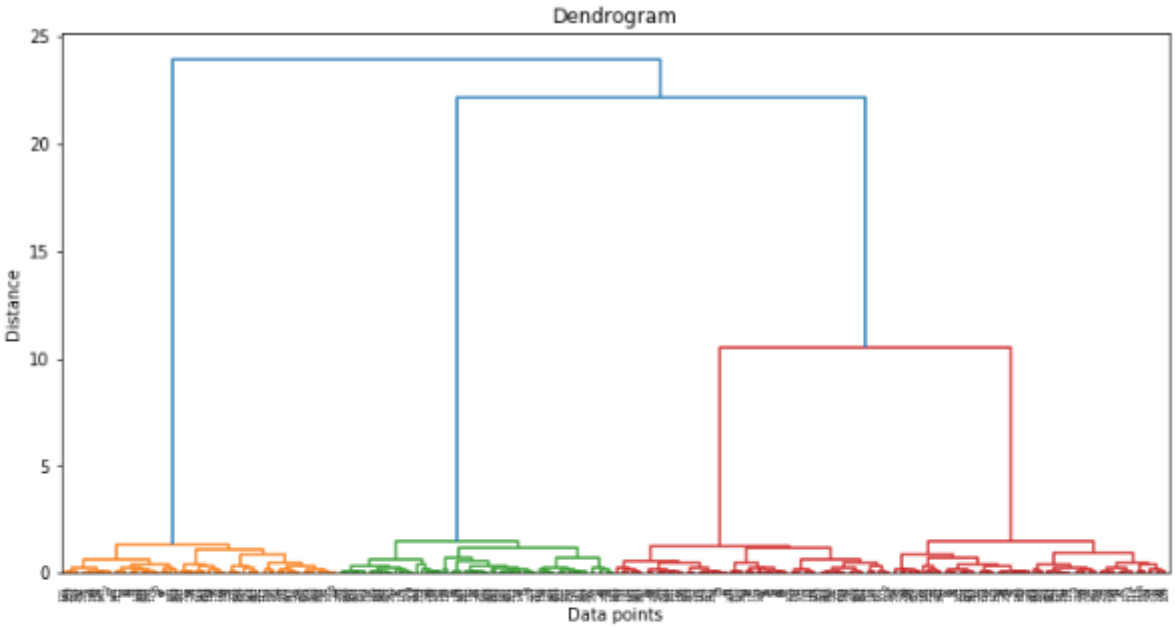


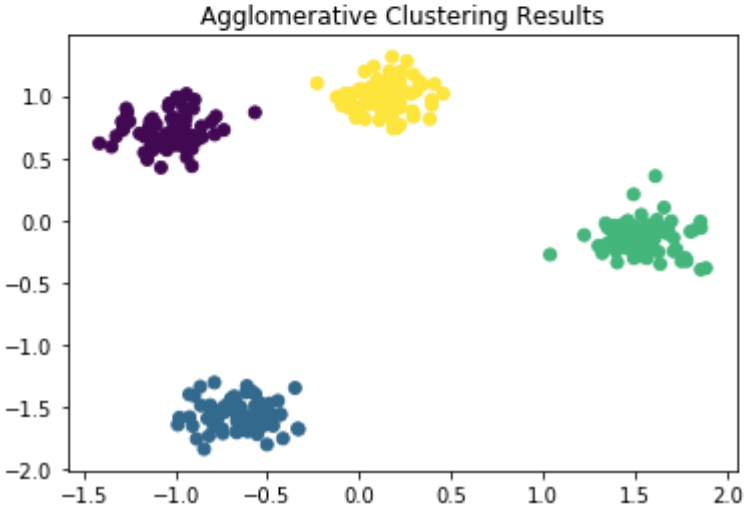
2. 계층적 군집 (Hierarchical Clustering)

- **연결(Linkage) 방법**: 군집 간의 거리를 측정하는 방법으로, `ward`(분산을 최소화), `complete`(최대 거리), `average`(평균 거리) 등이 있습니다.

```
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering

# 1. 덴드로그램 시각화
# linkage_matrix = linkage(데이터, method='연결방법')
linkage_matrix = linkage(X_scaled, method='ward')

plt.figure(figsize=(12, 6))
dendrogram(linkage_matrix)
plt.title('Dendrogram')
```

```
plt.xlabel('Data points')
plt.ylabel('Distance')
plt.show()

# 2. AgglomerativeClustering 모델 학습
# 덴드로그램을 보고 적절한 군집 수(n_clusters)를 결정
agg_cluster = AgglomerativeClustering(n_clusters=4, affinity='euclidean',
linkage='ward')
agg_labels = agg_cluster.fit_predict(X_scaled)

# 3. 결과 시각화
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=agg_labels, cmap='viridis',
marker='o')
plt.title('Agglomerative Clustering Results')
plt.show()
```

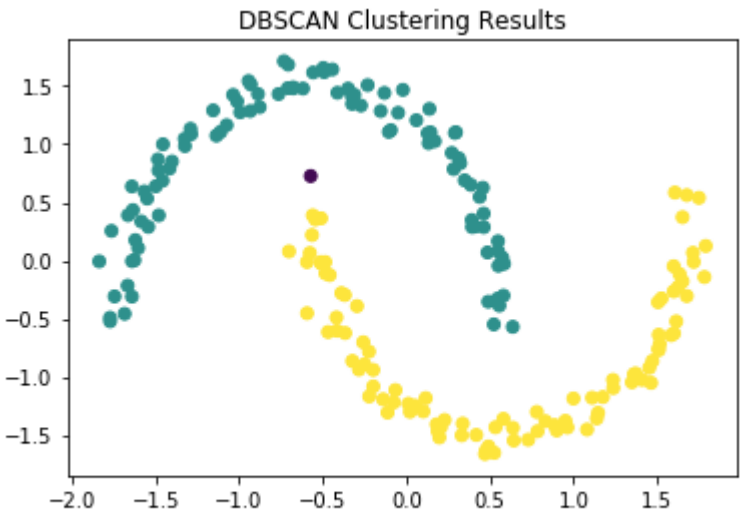




3. DBSCAN

- **하이퍼파라미터 튜닝**: `eps`와 `min_samples` 값에 따라 군집 결과가 크게 달라지므로, 데이터의 특성을 고려하여 적절한 값을 찾는 것이 중요합니다.

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_moons

# 1. 데이터 생성 (복잡한 형태)
X_moon, y_moon = make_moons(n_samples=200, noise=0.05, random_state=42)
X_moon_scaled = StandardScaler().fit_transform(X_moon)

# 2. DBSCAN 모델 학습
# eps: 이웃을 찾기 위한 거리
# min_samples: 군집을 형성하기 위한 최소 샘플 수
dbscan = DBSCAN(eps=0.3, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_moon_scaled) # -1은 노이즈(이상치)를 의미

# 3. 결과 시각화
plt.scatter(X_moon_scaled[:, 0], X_moon_scaled[:, 1], c=dbscan_labels,
cmap='viridis', marker='o')
plt.title('DBSCAN Clustering Results')
plt.show()

print("군집 레이블 종류:", np.unique(dbscan_labels)) # [-1  0  1]
```




```
군집 레이블 종류: [-1  0  1]
```

군집 성능 평가

비지도 학습인 군집 분석은 정답이 없으므로, 주로 **실루엣 계수(Silhouette Coefficient)**를 사용하여 성능을 평가합니다.

- **실루엣 계수**: 각 데이터 포인트가 자신이 속한 군집과 얼마나 유사하고, 다른 군집과는 얼마나 다른지를 측정한 값입니다.
  - +1에 가까울수록: 다른 군집과 잘 분리되어 있고, 자신의 군집 내에서는 밀집되어 있음을 의미.
  - 0에 가까울수록: 군집의 경계에 위치함을 의미.
  - -1에 가까울수록: 잘못된 군집에 할당되었음을 의미.
- 전체 데이터의 실루엣 계수 평균값이 1에 가까울수록 군집화가 잘 되었다고 평가합니다.

```python
from sklearn.metrics import silhouette_score, silhouette_samples

# K-Means 결과로 실루엣 계수 계산
score = silhouette_score(X_scaled, cluster_labels)
print(f"K-Means Silhouette Score: {score:.3f}") # 0.839
```