BDA x 이지스 퍼블리싱 머신러닝 스터디 5주차 과제

조 이름: 01조 (김예진, 김지웅, 박효정, 편민우, 홍소은)

1. K-평균 군집화 모델의 평가지표인 Rand 지수와 실루엣 계수에 대해 설명해 보시오.

- Rand 지수 (Rand Index, RI)

랜드 지수는 가능한 모든 데이터 쌍의 개수에 대해 정답인 데이터 쌍의 개수의 비율로 정의한다.

$$Rand\ Index = \frac{a+b}{{}_{N}C_{2}}$$

랜드 지수는 0에서 1까지의 값을 가지고, 1이 가장 좋은 성능을 뜻한다. 랜드 지수의 문제점은 무작위로 군집화를 한 경우에도 어느 정도 좋은 값이 나올 가능성이 높다는 점이다. 즉, 무작위 군집화에서 생기는 랜드 지수의 기댓값이 너무 크다. 이를 해결하기 위해 무작위 군집화에서 생기는 랜드 지수의 기댓값을 원래의 값에서 빼서 기댓값과 분산을 재조정한 것이 조정 랜드 지수(adjusted Rand index, ARI)이다.

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

조정 랜드 지수는 성능이 완벽한 경우 1이 된다. 반대로 가장 나쁜 경우로서 무작위 군집화를 하면 0에 가까운 값이 나온다. 경우에 따라 음수가 나올 수도 있다.

- 실루엣 계수 (silhouette coefficient)

개별 데이터가 할당된 군집 내 데이터와 얼마나 가깝게 군집화 되어 있는지, 그리고 다른 군집에 있는 데이터와는 얼마나 멀리 분리되어 있는지 나타내는 수치

군집이 잘 분리되었다는 것은 동일 군집 내에서의 데이터는 서로 가깝게 위치해 있으며, 다른 군집과의 거리는 멈을 의미한다. 즉 값이 크다면 군집화의 성능이 좋다고 해석한다.

실루엣 계수는 아래의 식에 의해 계산된다.

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

이때, a(i)는 개별 데이터의 동일 군집 내 다른 데이터들과의 평균 거리, b(i)는 가장 가까운 군집과의 평균 거리

실루엣 계수는 -1에서 1 사이의 값을 가지며, 1에 가까울수록 근처 군집과 멀리 떨어져 있음을, 0에 가까울수록 근처 군집과 가까움을, 마이너스이면 아예 다른 군집에 할당되었음을 의미한다.

각 데이터에서 실루엣 계수를 구한 후, 평균을 내면 전체 데이터의 실루엣 스코어를 구할 수 있다.

일반적으로 이 값이 크면(1에 가까우면) 군집화가 잘 되었다고 판단한다. 하지만 단순히 값이 크다고 해서 군집화가 잘 되었다고 판단할 수는 없다.

전체 실루엣 스코어와 개별 군집 평균값의 편차가 크지 않은 경우에 전체 군집화 성능이 좋다고 판단한다. 즉, 개별 군집의 실루엣 스코어가 전체 실루엣 스코어와 크게 다르지 않아야 한다.

실루엣 계수의 장점

1. 식이 단순하고 직관적이어서 이해하기 쉽다.
2. 군집 개수를 다르게 하여 각 값을 비교함으로써 최적의 군집 개수를 정하는 데 사용할 수 있다.

실루엣 계수의 단점

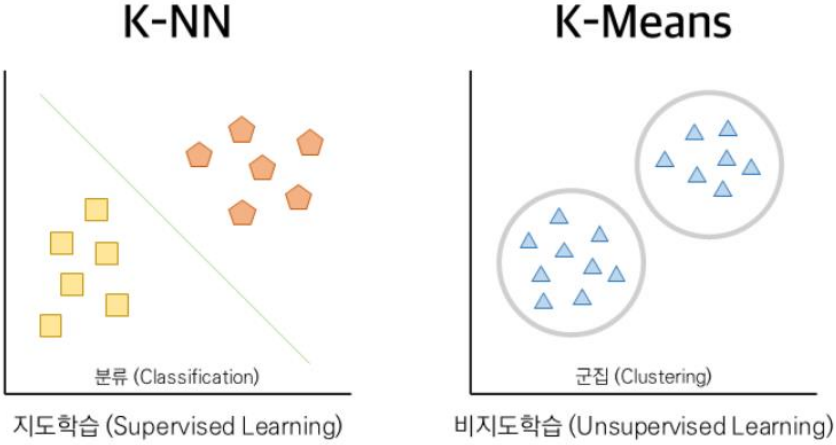
1. 밀도 기반 클러스터링 알고리즘(밀도가 높은 부분을 클러스터링하는 방식, 대표적으로 DBSCAN 알고리즘 이 있다.)에 한해 값이 크게 산출되는 경향이 있기 때문에, 유의해야 한다. 즉, 중심 기반 알고리즘을 사용해 클러스터링 한 후 비교하게 도면, 밀도 기반 알고리즘이 유리하게 나타날 가능성이 높다.
2. 데이터마다 다른 데이터와의 거리를 반복적으로 계산해야 되기 때문에 데이터가 많으면 계산량이 급격히 늘어

난다. 이런 경우, 군집별로 데이터를 샘플링하여 평가하는 방안을 모색해야 한다.

---

2. K-최근접 이웃 모델과 K-평균 군집화 모델의 차이를 서술하고, 각 모델이 활용될 수 있는 사례를 적어보시오.

	K-최근접 이웃(KNN) 모델	K-평균(K-means) 군집화 모델
유형	지도학습	비지도학습
알고리즘 목표	분류(classification) 및 회귀(regression) : 새로운 데이터 포인트의 클래스 레이블 예측	군집화(clustering) : 군집화된 데이터 포인트들의 데이터셋 안에서 패턴을 찾음
레이블(정답) 유무	O : 미리 레이블이 주어진 데이터를 학습하여 이를 바탕으로 새로운 데이터에 대해 분류 수행	X : 레이블이 주어지지 않았을 때 비슷한 특징을 가진 데이터끼리 묶는 군집 수행
k의 의미	분류하고자 하는 데이터와 거리상 근접한 이웃의 개수	클러스터링하고자 하는 군집의 개수
방식	해당 데이터와 가장 가까이 있는 k개의 데이터를 확인해 새로운 데이터 특성을 확인	점들 간 거리를 계산하는 metric의 결과값에 따라 k개의 중심점을 주변으로 데이터 포인트 배치
입력파라미터	최근접 이웃의 개수 선택 필요	군집의 개수를 필요로함
활용 사례	- 이미지 인식 - 음성 및 필기 인식 - 의료 진단 - 추천 시스템 - 신용 점수 - 품질 관리 - 유전자 발현 분석 - 자연어 처리(NLP)	- 이미지 분할 - 문서 분류 - 고객 세분화 - 사이버 프로파일링 - 은행 및 보험 분야 사기 탐지

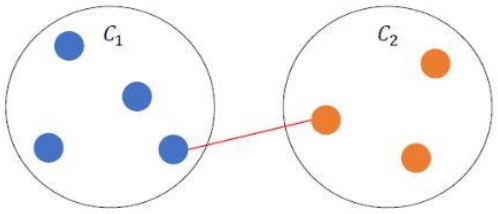


3. 계층적 군집화 모델에서 군집 사이의 거리를 정의하는 연결법들을 나열해보고 각 연결법들의 특징에 대해 서술해보시오.

계층적 군집화에서는 여러 가지 연결법을 통해 군집 간의 거리를 정의하게 되는데, 대표적인 연결법에는 최단 연결법(single linkage), 최장 연결법(complete linkage), 평균 연결법(average linkage), 중심 연결법(centroid linkage) 그리고 와드 연결법(Ward's linkage)이 있다.

1) 최단 연결법 : 가장 가까이에 있는 샘플간의 거리를 군집간의 거리로 보는 법

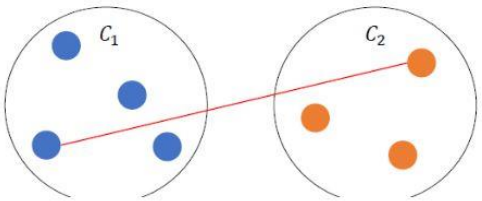
- $\min_{(x_1 \in C_1, x_2 \in C_2)} dist(x_1, x_2)$



- 이상치에 민감
- 계산량 많은 편

2) 최장 연결법 : 가장 먼 샘플간의 거리를 군집간의 거리를 보는 법

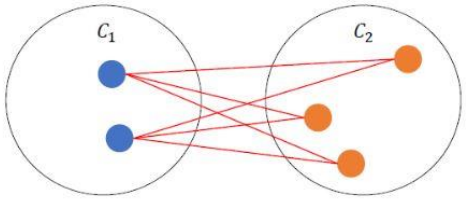
- $\max_{(x_1 \in C_1, x_2 \in C_2)} dist(x_1, x_2)$



- 이상치에 민감
- 계산량 많은 편

3) 평균 연결법 : 전체 거리에 대한 평균 거리를 군집간의 거리로 보는 법

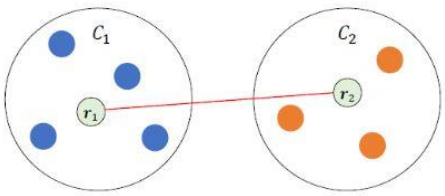
- $\frac{\sum_{x_1 \in C_1} \sum_{x_2 \in C_2} dist(x_1, x_2)}{|C_1| \times |C_2|}$



- 이상치에 둔감
- 계산량 많은 편

4) 중심 연결법 : 각 군집에 대해 중심을 구하고 중심간의 거리 계산

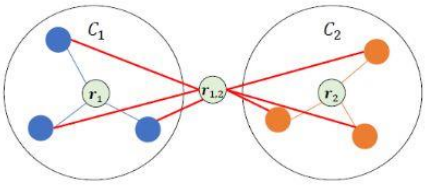
- $dist(r_1, r_2)$, 여기서 r_1과 r_2는 C_1과 C_2의 중심 $(r_1 = mean_{x_1 \in C_1}(x_1), r_2 = mean_{x_2 \in C_2}(x_2))$



- 이상치에 둔감
- 계산량 적은 편

5) 와드 연결법 : c1,c2를 하나의 군집으로 보았을 때의 중심과 각 중심과의 거리를 구하는 방식

- $\sum_{x_1 \in C_1} dist(x_1, r_1) + \sum_{x_2 \in C_2} dist(x_2, r_2) - \sum_{x \in C_1 \cup C_2} dist(x_1, r_{1,2})$



- 이상치에 매우 둔감
- 계산량 매우 많은 편
- 군집 크기 비슷하게 만듦

---

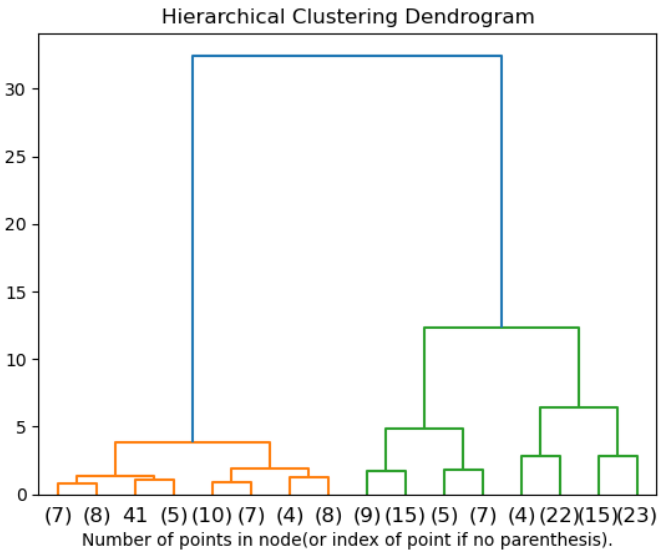
4. 교재 346페이지에서 덴드로그램 시각화 예제를 실습해보면서 각 과정에 대해 설명해보시오.

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from sklearn.datasets import load_iris
from sklearn.cluster import AgglomerativeClustering

# 붓꽃 데이터셋을 X에 저장
X = load_iris().data
```

```
# 계층적 군집화 모델 생성하고, 거리 임계값을 0으로 설정하여 훈련
model = AgglomerativeClustering(distance_threshold=0, n_clusters=None)
model = model.fit(X)
```

```
# 각 노드의 자식 노드 정보와 거리, 자식 노드에 속한 샘플 수 저장할 배열 초기화
counts = np.zeros(model.children_.shape[0])
n_samples = len(model.labels_)

# 각 노드에 대한 정보를 순회하면서 자식 노드의 샘플 수 계산하여 counts 배열 업데이트
for i, merge in enumerate(model.children_):
    current_count = 0
    for child_idx in merge:
        if child_idx < n_samples:
            current_count += 1
        else:
            current_count += counts[child_idx - n_samples]
        counts[i] = current_count

# linkage_matrix 생성
linkage_matrix = np.column_stack([model.children_, model.distances_, counts]).astype(float)
```

```
# 덴드로그램 생성시 레벨에 맞게 자르고, 최대 3개의 레벨만 표시
dendrogram(linkage_matrix, truncate_mode = "level", p=3)

# 그래프의 제목과 X축 레이블 지정
plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("Number of points in node(or index of point if no parenthesis).")
plt.show() # 그래프 표시
```



5. PCA 모델과 릿지 회귀에는 어떠한 연관성이 있는지 설명해 보이시오.

릿지 회귀의 해를 X의 특잇값 분해를 사용하여 나타낼 때, 규제가 없는 최소제곱법 모델의 해가 이미 존재한다는 것은 Ridge 회귀와 최소제곱법 모델의 해가 동일하다는 것을 의미하는데. 이 경우에는 Ridge 회귀에서의 규제항이 없어진 것으로 간주할 수 있다.

이를 통해 PCA와 Ridge 회귀 간의 관련성을 보면, PCA는 주성분 분석이라고도 불리며, 고차원 데이터의 차원을 줄이기 위해 주요한 특성을 추출하는 기술임으로. PCA의 목표는 데이터의 분산을 최대한 보존하는 새로운 축(주성분)을 찾는 것이다.

PCA의 수학적 기반 중 하나가 특잇값 분해인데, 특잇값 분해를 통해 데이터 행렬을 고유벡터와 고유값으로 분해할 수 있다. Ridge 회귀에서도 특잇값 분해를 사용하여 해를 구할 수 있는 특징이 있다.

따라서 규제가 없는 최소제곱법 모델의 해와 Ridge 회귀의 해가 동일하다는 것은, 데이터의 주성분을 찾는 PCA와 Ridge 회귀 간에 유사성이 있다는 것을 나타내고, Ridge 회귀는 회귀 모델에 L2 규제를 추가하여 과적합을 방지하고 안정성을 높이는 데 사용된다. 이와 관련하여 Ridge 회귀의 해를 특잇값 분해로 표현할 수 있다면, 이는 데이터의 주성분을 고려한 회귀 방법으로 해석될 수 있다.

---

6. 차원 축소기법인 MDS, Isomap, LLE, t-SNE의 특징에 대해 서술해보시오.

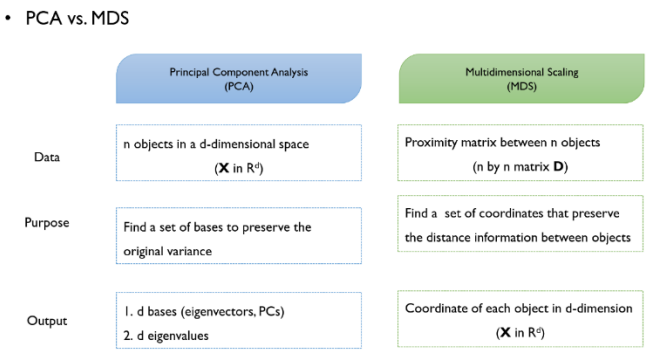
- MDS (Multidimensional Scaling)

MDS는 다차원 척도법이라고도 하며, 각 데이터 포인트 간의 거리를 보존하려고 하는 차원 축소 방법으로. MDS는 원래 고차원에서의 객체 간 거리와 저차원(일반적으로 2차원 또는 3차원)에서의 거리가 최대한 비슷하도록 점들을 배치한다.

In [1]:
```
from sklearn.datasets import load_digits
from sklearn.manifold import MDS

# digits 데이터 로드
digits = load_digits()
X = digits.data

# MDS 객체 생성
mds = MDS(n_components=2)

# MDS 적용
X_reduced = mds.fit_transform(X)
```

In [2]:
```
print(X_reduced)
```

```
[[-13.68918592  26.53981587]
 [ -3.89451729 -30.13116721]
 [-11.58717063 -30.14311115]
 ...
 [-11.49632164 -14.11273693]
 [ -6.36426888  18.11496109]
 [ -7.63199017   7.79690095]]
```



- Isomap (Isometric Mapping):

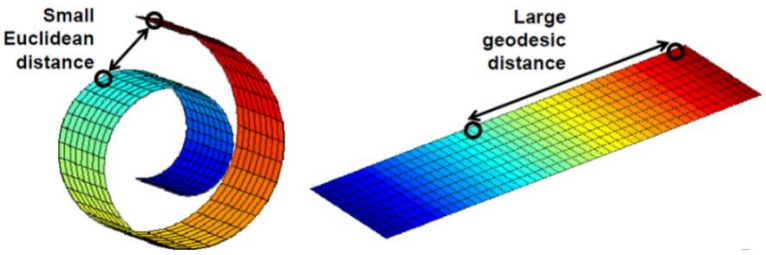
Isomap은 MDS를 확장한 방법으로, 각 데이터 포인트 간의 지오데식 거리를 보존하려고 하며. 데이터의 매니폴드를 이해하는 데 유용하고, 이를 통해 고차원 데이터의 내재적인 구조를 파악하는 데 도움을 준다.

In [6]:
```
from sklearn.datasets import load_digits
from sklearn.manifold import Isomap

# digits 데이터 로드
digits = load_digits()
X = digits.data

# Isomap 객체 생성
isomap = Isomap(n_components=2, n_neighbors=5)

# Isomap 적용
X_reduced_isomap = isomap.fit_transform(X)
```

```
C:₩Users₩kgw08₩anaconda3₩lib₩site-packages₩skle
rs graph is 2 > 1. Completing the graph to fit
  self._fit_transform(X)
C:₩Users₩kgw08₩anaconda3₩lib₩site-packages₩scip
atrix is expensive. lil_matrix is more efficien
  self._set_intXint(row, col, x.flat[0])
```

In [9]:
```
print(X_reduced_isomap)
```

```
[[163.39526445  28.06891135]
 [-46.00448978  48.29792057]
 [-97.23256557  21.60527119]
 ...
 [-49.97218081 -24.93724342]
 [ -0.96914751 -71.60775029]
 [ -9.41416423 -36.8662784 ]]
```



- LLE (Locally Linear Embedding)

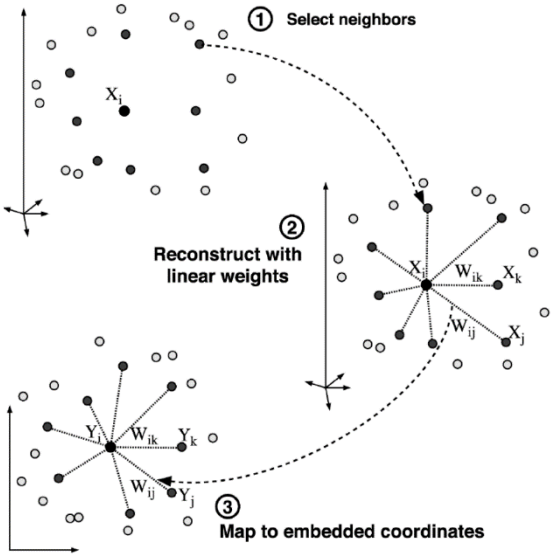
LLE는 각 데이터 포인트의 국소적인 선형 관계를 보존하려고 하며. 데이터의 매니폴드를 '풀어헤치는' 데 특히 유용하다. LLE는 비선형 구조를 가진 데이터에 대해 뛰어난 성능을 보이지만, 국소적인 정보를 기반으로 하기 때문에 전역적인 구조를 잘 잡아내지 못하는 한계가 있다

In [7]:
```
from sklearn.datasets import load_digits
from sklearn.manifold import LocallyLinearEmbedding

# digits 데이터 로드
digits = load_digits()
X = digits.data

# LLE 객체 생성
lle = LocallyLinearEmbedding(n_components=2, n_neighbors=5)

# LLE 적용
X_reduced_lle = lle.fit_transform(X)
```

In [10]:
```
print(X_reduced_lle)
```

```
[[-0.06778396  0.00579765]
 [ 0.00242334 -0.02744543]
 [ 0.00173576 -0.02712232]
 ...
 [ 0.00118942 -0.0268627 ]
 [-0.0025903  -0.02507498]
 [-0.00230939 -0.02520585]]
```

$$\Phi(W) = \sum_i \left| y_i - \sum_j W_i k y_j \right|^2$$



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

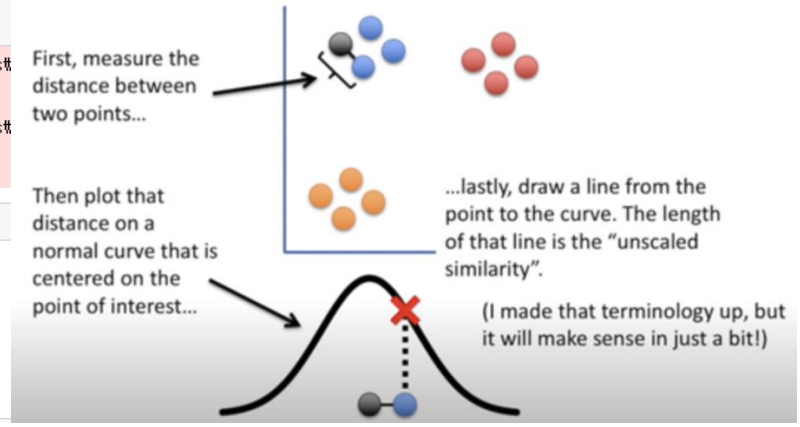
t-SNE는 동일한 클러스터나 그룹에 속한 데이터 포인트들이 저차원에서도 가까이 위치하도록 하는 방법으로. 고차원 데이터의 시각화에 유용하다. 하지만 t-SNE는 계산 복잡도가 높고, 하이퍼파라미터에 민감하다는 단점이 존재한다.

In [8]:
```
from sklearn.datasets import load_digits
from sklearn.manifold import TSNE

# digits 데이터 로드
digits = load_digits()
X = digits.data

# t-SNE 객체 생성
tsne = TSNE(n_components=2)

# t-SNE 적용
X_reduced_tsne = tsne.fit_transform(X)
```

```
C:₩Users₩kgw08₩anaconda3₩lib₩site-packages₩
rom 'random' to 'pca' in 1.2.
  warnings.warn(
C:₩Users₩kgw08₩anaconda3₩lib₩site-packages₩
om 200.0 to 'auto' in 1.2.
  warnings.warn(
```

In [11]:
```
print(X_reduced_tsne)
```

```
[[ 6.2812244e+01 -1.5875430e+01]
 [-2.1426706e+01 -2.1313494e+01]
 [-2.9392338e+01 -3.8184887e-01]
 ...
 [-1.8122263e+01 -6.2679909e-02]
 [-1.1397141e+01  3.4437286e+01]
 [-1.9155827e+01  6.3945951e+00]]
```



	대분류	중분류	Example	Class label	Algorithms
Dimensionality Reduction 차원축소	Feature Selection 변수선택	Filter (un-supervised)	• Information Gain (IG) • Odds Ratio	Used	Not Used
		Wrapper (supervised)	• Forward Selection • Backward Selection • Stepwise Selection • Genetic Algorithm	Used	Used
	Feature Extraction 변수추출	Max Variance	• Principal Component Analysis (PCA)	Not Used	Not Used
		Max Distance Info.	• Multidimensional Scaling (MDS)	Not Used	Not Used
		Reveal non-linear structure	• LLE • ISOMAP • t-SNE	Not Used	Not Used