7.1 K-평균(K-Means) 알고리즘 정리

1. K-평균이란?

: **군집화(Clustering)** 알고리즘 중 가장 널리 쓰이는 비지도학습 알고리즘
데이터를 K개의 그룹(cluster)으로 나누고 각 군집의 중심값을 기준으로 가장 가까운 데이터들을 묶는 방식.

2. 알고리즘 핵심 아이디어

- 군집 중심 = 해당 군집에 속한 데이터들의 **평균 위치**
- 중심을 계속 업데이트하며 반복적으로 군집을 재조정
- 중심 좌표 이동이 더 이상 발생하지 않으면 종료

3. 동작 과정 (반복 학습 구조)

1. K개의 군집 중심을 **임의로 설정**
2. 각 데이터를 가장 가까운 군집 중심에 할당
3. 군집에 속한 데이터들의 **평균 위치로 중심 이동**
4. 중심이 변하면 다시 2~3 반복
5. 더 이상 중심 변화가 없으면 종료

4. 중요한 하이퍼파라미터

- n_clusters : 군집 개수 (가장 중요)
- init : 초기 중심 설정 방식 (기본: 'k-means++')
- max_iter : 반복 횟수 제한
- random_state : 난수 시드

5. 주요 속성

- labels : 각 데이터의 군집 라벨
- cluster_centers : 각 군집 중심 좌표
- inertia_ : 군집 내 오차합(SSE)

6. K-평균의 장단점

장점 : 가장 널리 사용됨, 구현 간단 & 빠름, 대용량 데이터에서도 효율적

단점 : 군집 수 K를 미리 정해야 함, 군집이 원형(구형) 형태일 때만 좋은 성능, 이상치에 매우 민감, 초기 중심 위치에 따라 결과가 달라짐

*** 실루엣 계수**

: 군집이 얼마나 분리되어 있는지를 판단하는 지표

- 1에 가까울수록 잘 분리된 군집
- 0이면 경계 모호
- 음수면 잘못 군집됨

 # KMeans 코드 핵심 예시

```
from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

from sklearn.preprocessing import StandardScaler

import numpy as np

import pandas as pd


X_scaled = StandardScaler().fit_transform(X)


kmeans = KMeans(n_clusters=3, random_state=0)

labels = kmeans.fit_predict(X_scaled)


print("silhouette score:", silhouette_score(X_scaled, labels))
```

 # 로그 변환 + 스케일링 예시

```
cust_df['Recency_log'] = np.log1p(cust_df['Recency'])
cust_df['Frequency_log'] = np.log1p(cust_df['Frequency'])
cust_df['Monetary_log'] = np.log1p(cust_df['Monetary'])

X = cust_df[['Recency_log', 'Frequency_log', 'Monetary_log']].values
X_scaled = StandardScaler().fit_transform(X)
```

7. KMeans 결과 해석

- labels → 각 데이터가 어느 cluster인지
- cluster_centers → cluster의 중심 위치
- groupby로 cluster 특징 비교 가능

```
cust_df.groupby("cluster_label")['Recency'].mean()
```

*** PCA로 군집 시각화**

KMeans는 고차원에서도 작동하지만 시각화를 위해 PCA로 2차원 축소 가능

7.2 군집 평가 Cluster Evaluation

군집 평가가 필요한 이유

- 지도학습과 달리 대부분의 군집 데이터는 레이블이 없음
- 군집이 잘 되었는지 평가하기 어려움 → 거리 기반 내부 지표 필요
- 군집의 품질을 정량적으로 평가하기 위해 가장 널리 쓰이는 방법이 실루엣 분석

1. 실루엣 분석 (Silhouette Analysis) 개요

*a(i) : 같은 군집 내 다른 데이터들과의 평균 거리

*b(i) : 가장 가까운 다른 군집의 평균 거리

>> 내 군집 내부에서는 가까워야 하고(a), 다른 군집과는 멀어야 한다(b)가 이상적 군집

* 실루엣 계수 공식 : $s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$

2. 실루엣 계수 범위 해석

- 1에 가까움 : 군집이 잘 분리됨, 이상적

- 0 근처 : 군집 경계에 위치

- 음수 : 잘못된 군집에 배정됨.

sklearn 실루엣 함수

from sklearn.metrics import silhouette_samples, silhouette_score

- silhouette_samples() : 개별 데이터의 실루엣 계수 반환

- silhouette_score() : 전체 평균 실루엣 계수 반환

3. 실루엣 분석으로 "좋은 군집"이 되려면

1. 전체 평균 실루엣 계수가 높을수록 좋다 (0~1)
2. 군집별 실루엣 계수의 편차가 크지 않아야 한다
3. 모든 군집 내 값이 균일해야 한다
   → 한 군집만 잘되고 나머지가 엉망이면 나쁜 결과

4. 군집 개수 결정(최적 k 찾기) 방법

➞ 군집 개수를 여러 개 설정해 보고 평균 실루엣 계수 변화를 비교

➞ visualize_silhouette() 같은 그래프를 통해 시각적으로 판단

*** 예시 비교**

군집 개수	평균 실루엣 점수	해석
2	0.704	매우 좋음, 하지만 한 군집이 불균형
3	0.588	전체적으로 개선 X
4	0.651	가장 안정적이며 군집별 균형도 좋음
5	0.575	오히려 더 나빠짐

>> 평균점수 + 군집별 균형 고려했을 때 **4개가 최적 군집 수**

5. 결론

*** 좋은 군집**

✓ 군집 내부 밀집도 ↑
✓ 군집 간 분리도 ↑
✓ 실루엣 score ≥ 0.6 이상이면 매우 좋은 편
✓ 군집별 균형 유지 (편차 낮음)

*** 나쁜 군집**

✗ 실루엣 계수 음수
✗ 평균은 높지만 일부 군집이 극단적으로 낮음
✗ 군집 수가 많을수록 항상 좋은 것은 아님

7.3 평균 이동 군집화 요약

1. 개요

평균 이동 : **비지도 학습 기반의 군집 알고리즘**, 군집 개수를 미리 지정할 필요 없음..
데이터가 가장 밀집된 방향으로 중심점이 이동하며 결과적으로 데이터의 최고 밀도 위치를 군집 중심 찾는다.

* 핵심 개념: 확률 밀도 함수(PDF) + KDE

2. Mean Shift의 핵심 개념

*** KDE (커널 밀도 추정)**

- 데이터 분포를 부드럽게 추정하는 확률 밀도 함수
- 가우시안 커널을 가장 많이 사용
- 커널 크기 = **대역폭**

*** bandwidth(h)의 의미**

- **너무 작음 :** 과적합 (군집이 너무 많이 생김)
- 너무 큼 : 과소적합 (군집이 너무 적게 생김, 뭉개짐)

3. 알고리즘 동작 과정

1. 각 데이터 포인트 근처 밀도 계산 (KDE)
2. 가장 밀도가 높은 방향으로 중심을 이동
3. 이동이 수렴할 때까지 반복
4. 최종 중심 위치가 군집 중심이 됨

4. Mean Shift vs K-Means

비교항목	Mean Shift	K-Means
군집 개수	자동 결정	지정 필요
군집 형태	아무 형태 가능	구형(원형) 가정
이상치 영향	작음	큼
연산 속도	느림	빠름

7.4 GMM 요약 정리

1. GMM이란?

- 데이터가 여러 개의 정규분포(가우시안 분포)가 섞여 있다고 가정하고 군집화하는 방식 + 거리 기반이 아닌 **확률 기반** 군집화
- 데이터가 단일 중심이 아닌 **비대칭, 타원형, 다양한 분포 형태**라도 잘 처리 가능

2. GMM의 학습 방법(EM 알고리즘)

단계	설명
E-step	각 데이터가 각 분포에 속할 확률을 계산
M-step	그 확률 기반으로 분포의 평균, 분산 등을 다시 추정

이를 반복하여 최적 매개변수를 찾음

3. GMM vs KMeans 비교

비교 항목	KMeans	GMM
군집 형태	원형/구형	타원형도 가능
군집 기준	거리	확률
속도	빠름	느림
유연성	낮음	매우 높음

>> GMM은 KMeans가 놓치는 타원형 군집도 잘 잡음

4. 실습 결과 비교

* KMeans 결과 : 오차 존재 VS GMM 결과 : Target과 더 잘 매핑됨

>> 군집 1번만 제대로 매핑된 KMeans에 비해 GMM은 전체 군집이 target과 매핑됨

>> **즉, GMM은 더 유연하고 복잡한 데이터에도 강함**

5. 핵심 요약

*** GMM의 장점**

- 다양한 형태의 데이터 분포에서 잘 동작
- 확률 기반이라 더 정교한 군집 결과
- 타원형, 불균등 분포 처리 가능

*** GMM의 단점**

- EM 반복으로 인해 **속도 느림**
- 파라미터 수가 많아 복잡함

6. GMM 사용 환경

- 군집이 둥근 모양이 아닐 때
- 데이터에 여러 분포가 섞여 있을 때
- soft clustering(확률 기반) 필요할 때
- KMeans가 잘 안 되는 데이터일 때

7.5 DBSCAN 개요

1. 핵심 특징

- 군집 기준 : 밀도 기반
- 형태 제약 없음 : 원형이 아닌 임의 모양의 군집도 탐지 가능
- 노이즈 처리 : 노이즈 자동 탐지
- 파라미터 : eps(ε), min_samples
- 장점 : 복잡한 분포 데이터도 잘 분류
- 단점 : eps, min_samples 튜닝이 중요 / 고차원에서는 어려움

2. 주요 파라미터

- eps (입실론)

→ 한 점을 기준으로 이웃으로 인정할 거리 반경

- min_samples (최소 데이터 개수)

→ Core Point이 되기 위해 점 주변에 포함돼야 하는 최소 개수
(자기 자신 포함 ⇒ min points = min_samples - 1)

3. DBSCAN에서 정의되는 4가지 데이터 유형

유형	조건	설명
핵심 포인트	eps 반경 내 데이터 ≥ min_samples	군집 중심
이웃 포인트	eps 내 포함되지만 Core는 아님	Core 주변 일반점
경계 포인트	ore 근처에 있으나 min_samples 미만	군집 경계
잡음 포인트	Core에도 Border에도 속하지 않음	군집 외부, 제거

4. 개념 흐름 요약

1. 일정 반경(eps) 내 최소 데이터 ≥ min_samples → Core Point
2. Core Points 끼리는 direct connection
3. 연결된 Core 영역 = 하나의 Cluster
4. Border Point는 군집 경계로 포함됨
5. 아무 데도 속하지 않는 점 = Noise

5. DBSCAN이 좋은 경우

- 군집이 비원형 / 복잡한 모양
- 밀도 차이가 뚜렷한 경우
- 노이즈가 포함된 경우

6. DBSCAN이 어려운 경우

- eps, min_samples 선택에 민감
- 데이터가 sparse하거나 고차원일 때 비효율적
- 밀도가 크게 다른 경우 잘 안 됨

7. KMeans vs DBSCAN vs GMM

알고리즘	군집 기준	모양	노이즈 처리
KMeans	중심+거리기반	원형	약함
GMM	확률 기반 (가우시안)	타원	약함
DBSCAN	밀도 기반	임의 형태 OK	굉장히 강함