

클러스터링과 이상 탐지

K-means · DBSCAN · Isolation Forest · LOF

레이블 없는 데이터에서 패턴과 이상값을 찾는 방법을 다룹니다.
PCA와 차원 축소 개념을 먼저 읽고 오면 더 이해하기 쉽습니다.

1. 들어가며: 레이블 없는 데이터에서 패턴 찾기

지금까지 이 시리즈에서 다룬 방법들(로지스틱 회귀, Random Forest, GBM)은 모두 '정답(레이블)'이 있는 훈련 데이터를 필요로 하는 지도학습(supervised learning)이었습니다. 그런데 현실에서는 레이블을 얻기 어렵거나 비용이 너무 큰 경우가 많습니다. 수백만 건의 거래 내역 중 이상한 것을 찾으려 할 때, 모든 거래를 사람이 직접 보고 '이상/정상'으로 레이블을 달기는 불가능합니다.

이 가이드는 레이블 없이도 데이터에서 의미 있는 구조를 찾는 두 가지 비지도학습(unsupervised learning) 기법을 다룹니다.

- 클러스터링: 비슷한 데이터끼리 그룹으로 묶습니다. '이 데이터에 몇 개의 패턴이 존재하는가?'라는 질문에 답합니다.
- 이상 탐지: 다른 데이터들과 눈에 띄게 다른 점들을 찾습니다. '이 데이터는 평소와 다른 무언가가 있는가?'라는 질문에 답합니다.

지도학습 vs 비지도학습: 결과 해석이 달라진다

지도학습: 정답이 있으므로 Accuracy, AUC 같은 명확한 성능 지표로 평가할 수 있습니다.

비지도학습: 정답이 없으므로 '얼마나 잘 됐는가'를 객관적으로 측정하기 어렵습니다.

이 가이드의 예제에서는 사후 검증용으로만 실제 레이블(악성/양성)을 사용합니다. 실제 적용 상황에서는 이 레이블이 없다고 가정해야 합니다.

예제 데이터셋

이 가이드는 시리즈 전체와 동일한 Breast Cancer Wisconsin 데이터셋(569 명, 30 개 변수)을 사용합니다. 클러스터링과 이상 탐지는 레이블 없이 수행하고, 실제 레이블(악성=1, 양성 종양=0 으로 뒤집어 사용)은 나중에 결과를 검증하는 데만 사용합니다. 또한 고차원 데이터를 2D 로 시각화하기 위해 이전 차원 축소 가이드에서 다룬 PCA 를 전처리로 활용합니다.

2. 클러스터링 기초

2-1. K-means: 중심까지의 거리로 군집 구성

K-means 는 가장 단순하고 널리 쓰이는 클러스터링 알고리즘입니다. 작동 방식은 다음과 같습니다.

1. 클러스터 수 k 를 먼저 정합니다.
2. k 개의 중심점(centroid)을 무작위로 초기화합니다.
3. 각 데이터 포인트를 가장 가까운 중심에 배정합니다.
4. 각 클러스터의 중심을 소속 데이터 포인트들의 평균으로 업데이트합니다.
5. 3~4 단계를 중심이 더 이상 움직이지 않을 때까지 반복합니다.

k 는 어떻게 정하는가

K-means 의 가장 큰 과제는 k 를 미리 정해야 한다는 것입니다. 두 가지 기준을 함께 씁니다.

- Elbow Method: 클러스터 내 거리 합(Inertia)을 k 별로 그렸을 때, 기울기가 꺾이는 'elbow' 지점이 적절한 k 의 힌트입니다.
- Silhouette Score: 각 포인트가 자신의 클러스터에 얼마나 잘 속하는지, 그리고 이웃 클러스터와 얼마나 잘 분리되는지를 $-1 \sim 1$ 사이로 측정합니다. 높을수록 좋습니다.

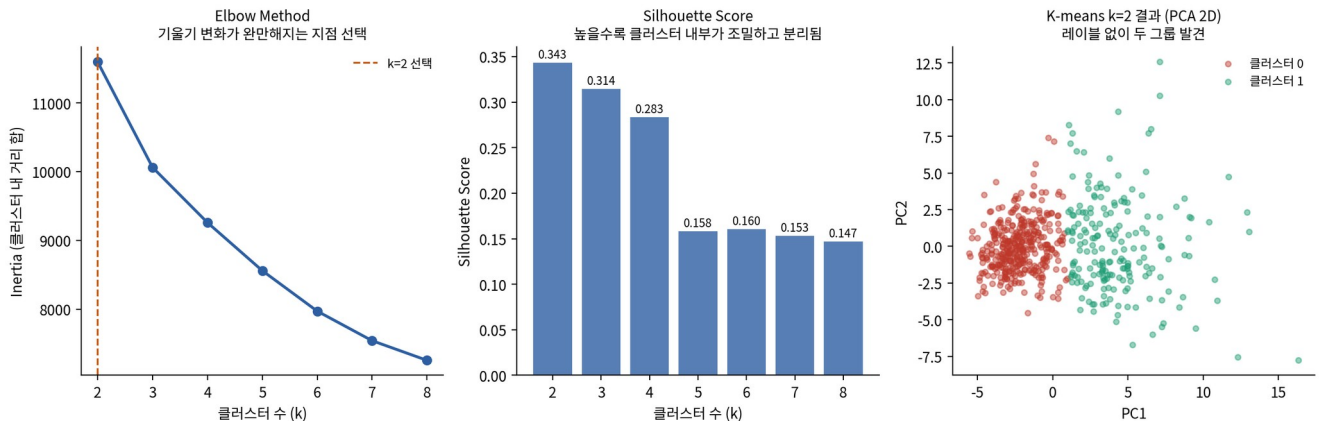


그림 1. K-means 클러스터 수 선택 기준. Elbow(왼쪽): $k=2 \sim 3$ 에서 기울기가 완만해진다. Silhouette(가운데): $k=2$ 가 가장 높아 (0.343) 두 그룹이 가장 잘 분리됨을 의미한다. K-means $k=2$ 결과(오른쪽): PCA 2D 에서 두 클러스터가 어느 정도 분리된다.

$k=2$ 에서 Silhouette(0.343)이 가장 높고, Elbow 에서도 $k=2 \sim 3$ 에서 꺾입니다. 실제로 이 데이터셋이 악성/양성 두 클래스로 구성되어 있음을 고려하면 합리적인 결과입니다. 그러나 현실에서는 이 검증을 할 수 없다는 점을 기억해야 합니다.

2-2. K-means 의 한계

- k 를 미리 알아야 합니다. 데이터에 자연적인 그룹이 몇 개인지 모르는 경우가 대부분입니다.
- 구형(spherical) 클러스터를 가정합니다. 클러스터 형태가 길쭉하거나 고리 모양이면 잘 작동하지 않습니다.
- 이상치에 민감합니다. 극단값이 있으면 중심점이 끌려가서 클러스터 품질이 떨어집니다.
- 초기화에 따라 결과가 달라질 수 있습니다. random_state 를 고정하거나 n_init 을 충분히 크게(기본값 10) 설정해야 합니다. 실제로 이 데이터에서 $n_init=10$ 으로는 inertia 11595.53, $n_init=100$ 으로는 11595.46 인 조금 더 나은 해가 나오며, 그에 따라 ARI 도 0.654 에서 0.671 로 달라집니다(R 의 kmeans()는 기본 설정으로 후자를 찾습니다).

2-3. R / Python 코드 (K-means)

R

```
library(stats)

set.seed(42)
# 반드시 표준화 후 적용
X_scaled <- scale(X)

# k=2로 K-means 실행
km_fit <- kmeans(X_scaled, centers=2, nstart=10)
print(table(km_fit$cluster)) # 클러스터 크기

# Silhouette Score
library(cluster)
sil <- silhouette(km_fit$cluster, dist(X_scaled))
print(mean(sil[,3])) # 평균 Silhouette

# k 선택: Elbow (Inertia = tot.withinss)
inertias <- sapply(2:8, function(k) {
  kmeans(X_scaled, centers=k, nstart=10)$tot.withinss
})
plot(2:8, inertias, type='b', xlab='k', ylab='Inertia')
```

Python

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import numpy as np

scaler = StandardScaler()
X_s = scaler.fit_transform(X) # 표준화 필수

# Elbow와 Silhouette로 k 탐색
for k in range(2, 9):
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = km.fit_predict(X_s)
    sil = silhouette_score(X_s, labels)
    print(f'k={k}: Inertia={km.inertia_:.1f}, Silhouette={sil:.4f}')

# 최적 k로 최종 클러스터링
km_final = KMeans(n_clusters=2, random_state=42, n_init=10)
labels = km_final.fit_predict(X_s)
```

2-4. DBSCAN: 밀도 기반, 노이즈 포인트 자동 탐지

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)은 K-means와 근본적으로 다른 방식으로 작동합니다. '밀도가 높은 지역은 클러스터고, 밀도가 낮은 지역에 있는 점은 노이즈다'라는 아이디어입니다.

두 파라미터가 핵심입니다.

- eps: 한 점의 이웃 반경. 이 반경 안에 있는 점들을 '이웃'으로 봅니다.

- min_samples: 한 점이 '핵심 점(core point)'이 되려면 eps 반경 안에 최소 몇 개의 이웃이 있어야 하는지. 이웃이 min_samples 보다 적은 점은 경계점이나 노이즈로 분류됩니다.

DBSCAN 의 가장 중요한 특성은 두 가지입니다. k 를 미리 정하지 않아도 되고, 이웃이 부족한 점을 자동으로 노이즈(-1)로 분류합니다. 이 노이즈 포인트들이 바로 이상치 탐지에 활용됩니다.

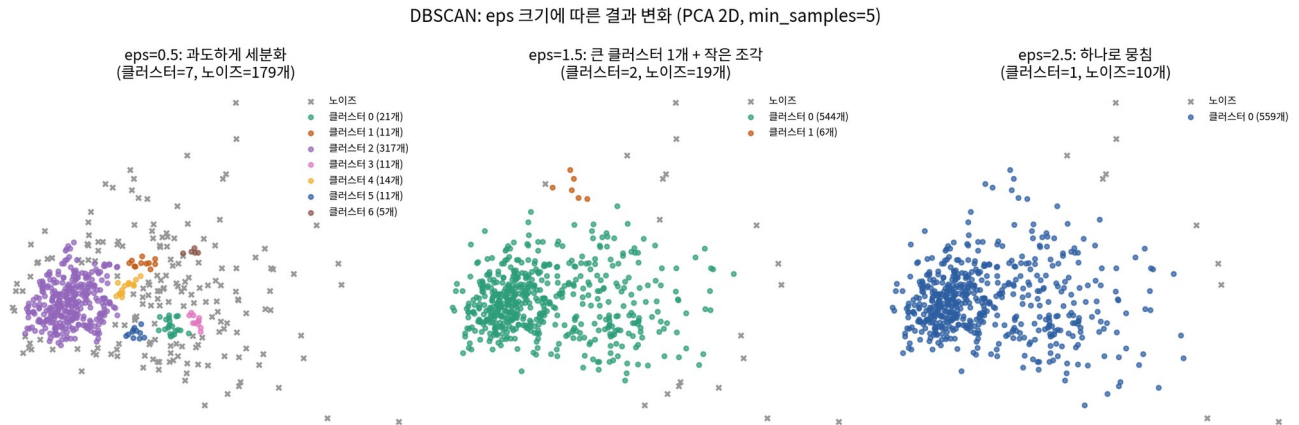


그림 2. DBSCAN 에서 eps 크기에 따른 결과 변화(PCA 2D, min_samples=5). eps 가 작으면 많은 점이 노이즈로 분류되며 클러스터가 과도하게 세분화된다. eps=1.5 에서는 클러스터가 2 개로 나오지만 544 개 대 6 개로, 사실상 큰 덩어리 하나와 작은 조각이다. eps 가 너무 크면 모든 점이 하나로 뭉친다. 이 데이터처럼 밀도가 뚜렷하게 나뉘지 않으면 DBSCAN 은 적절한 eps 를 찾기 어렵다.

2-5. R / Python 코드 (DBSCAN)

R

```
library(dbSCAN)

# PCA로 차원 축소 후 적용 권장
pca_fit <- prcomp(scale(X), center=FALSE)
X_pca <- pca_fit$x[, 1:2]

db_fit <- dbSCAN(X_pca, eps=1.5, minPts=5)

table(db_fit$cluster) # 0: 노이즈, 1,2,...: 클러스터
# 노이즈 포인트 추출
noise_idx <- which(db_fit$cluster == 0)
cat('노이즈 포인트 수:', length(noise_idx), '\n')
```

Python

```
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA

# PCA로 차원 축소 후 적용 권장
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X_s)

db = DBSCAN(eps=1.5, min_samples=5)
labels = db.fit_predict(X_pca)

n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
n_noise = (labels == -1).sum()
```

```
print(f'클러스터 수: {n_clusters}, 노이즈: {n_noise}개')
```

```
# 노이즈 포인트 = 이상치 후보  
noise_mask = labels == -1
```

3. 이상 탐지

3-1. 클러스터링 기반 이상 탐지 vs 전용 이상 탐지 기법

클러스터링 이상 탐지에 활용할 수 있습니다. K-means 에서 모든 클러스터 중심까지 거리가 먼 점, DBSCAN 에서 노이즈(-1)로 분류된 점이 이상치 후보입니다. 그러나 클러스터링은 본래 그룹을 찾는 것이 목적이므로 이상 탐지 전용 기법보다 정확도가 낮을 수 있습니다.

이상 탐지 전용 기법으로 이 가이드는 두 가지를 다룹니다.

3-2. Isolation Forest: 이상값은 고립시키기 쉽다

Isolation Forest 는 역발상에서 출발합니다. 정상 데이터는 다른 정상 데이터들과 비슷하므로 '고립'시키려면 많은 분기가 필요합니다. 반면 이상 데이터는 혼자 떨어져 있으므로 적은 분기만으로 고립됩니다.

Isolation Forest 는 무작위 결정 트리를 많이 만든 뒤, 각 데이터 포인트를 고립시키는 데 필요한 분기 수를 측정합니다. 분기 수가 적을수록 이상 점수가 높습니다. 실무에서 가장 먼저 정하는 파라미터는 contamination(이상치로 표시할 비율)입니다. 다만 contamination 은 이상 점수 자체를 바꾸지 않고 “점수 상위 몇 %를 이상치로 자를지”라는 임계값만 정합니다. 트리 수(n_estimators)와 트리당 샘플 수(max_samples)도 결과에 영향을 줍니다.

3-3. LOF: 주변 밀도와 비교

LOF(Local Outlier Factor)는 각 데이터 포인트 주변의 밀도를 인근 점들의 밀도와 비교합니다. 주변 밀도가 인근 점들보다 훨씬 낮으면 이상치입니다. 즉 '군중 속에 혼자 있는 점'을 찾습니다.

LOF 는 전역 기준이 아닌 지역 밀도를 기준으로 하므로, 밀도가 다른 여러 클러스터가 섞인 데이터에서 유리합니다. 다만 sklearn 에서 새 데이터에 LOF 를 적용하려면 LocalOutlierFactor(novelty=True)로 학습해야 합니다. 기본 설정에서는 fit_predict()로 학습 데이터 자체만 판정할 수 있습니다.

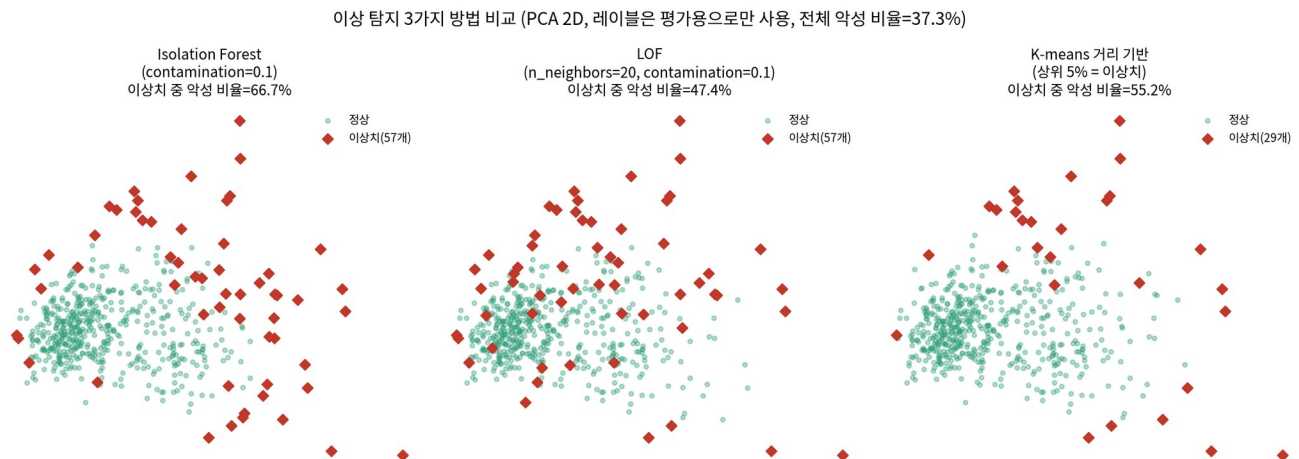


그림 3. 세 가지 이상 탐지 방법 비교(PCA 2D). 이상치(빨간 다이아몬드)의 위치와 분포가 방법마다 다르다. 전체 악성 비율 (37.3%)과 비교하면, Isolation Forest 가 이상치 중 악성 비율 66.7%로 가장 높았고 K-means 거리 기반 55.2%, LOF 47.4% 순이었다.

3-4. R / Python 코드 (이상 탐지)

R

```

library(isotree)
library(dbSCAN)

# Isolation Forest
iso_fit <- isolation.forest(X_scaled, ntrees=100,
                           sample_size=256)
iso_scores <- predict(iso_fit, X_scaled)
# 상위 10%를 이상치로
threshold <- quantile(iso_scores, 0.9)
anomalies_iso <- which(iso_scores > threshold)

# LOF
lof_scores <- lof(X_scaled, k=20)
threshold_lof <- quantile(lof_scores, 0.9)
anomalies_lof <- which(lof_scores > threshold_lof)

```

Python

```

from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

# Isolation Forest
iso = IsolationForest(
    contamination=0.1, # 이상치 비율 사전 지정
    random_state=42
)
iso_preds = iso.fit_predict(X_s) # -1=이상, 1=정상
iso_scores = iso.score_samples(X_s) # 낮을수록 이상

# LOF
lof = LocalOutlierFactor(
    n_neighbors=20,
    contamination=0.1
)
lof_preds = lof.fit_predict(X_s) # -1=이상, 1=정상

# 이상치 인덱스 추출
anomaly_idx = np.where(iso_preds == -1)[0]
print(f'이상치 {len(anomaly_idx)}개 탐지')

```

4. 평가의 어려움: 레이블이 없을 때

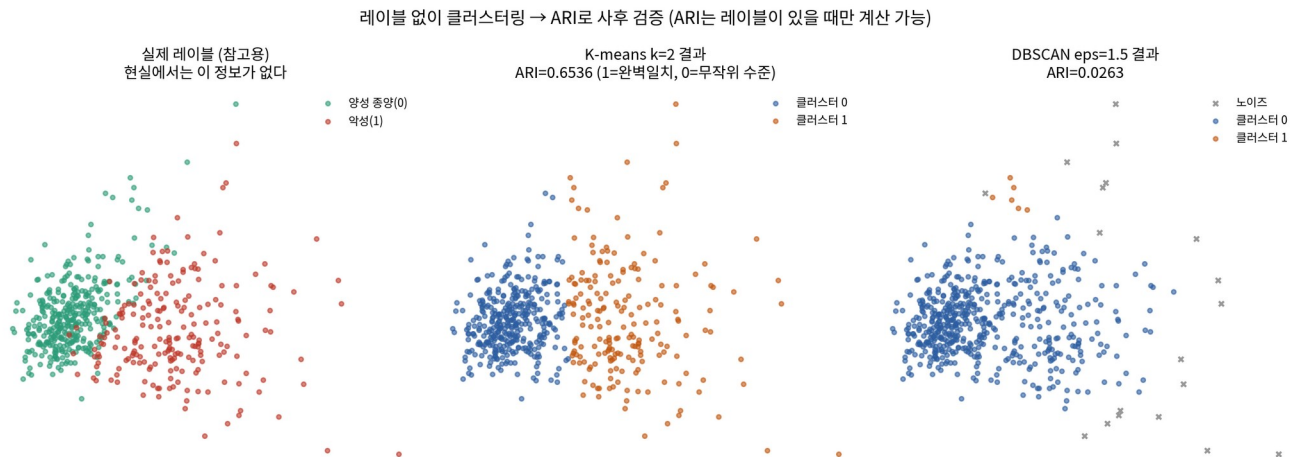


그림 4. 동일 데이터에 대한 세 가지 시각화. 실제 레이블(왼쪽)은 현실에서 없다고 가정해야 한다. K-means k=2(가운데)는 ARI=0.654로 실제 분류와 상당히 일치하지만, DBSCAN eps=1.5(오른쪽)는 ARI=0.026으로 이 데이터에서는 맞지 않았다. ARI 계산 자체가 레이블을 알아야만 가능하다.

비지도학습 결과를 평가하는 것은 지도학습보다 훨씬 어렵습니다.

4-1. 레이블이 없을 때 쓸 수 있는 지표

- Silhouette Score: 클러스터 내부 응집도와 클러스터 간 분리도를 측정. 레이블 없이 계산 가능. 다만 거리 기반 지표라 둥글고 볼록한 클러스터를 선호하므로, 고리 모양처럼 복잡한 형태의 클러스터에서는 실제 품질보다 낮게 나올 수 있습니다.
- Inertia / 왜곡도(distortion): K-means의 클러스터 내 거리 합. 낮을수록 좋지만, k가 커지면 무조건 낮아지므로 Elbow 형태로만 사용합니다.
- 도메인 지식: 클러스터 결과가 의미가 있는지 해당 분야 전문가가 직접 확인하는 것이 종종 가장 신뢰할 수 있는 평가 방법입니다.

4-2. 레이블이 있을 때 사후 검증

- ARI (Adjusted Rand Index): 두 분할(클러스터 결과 vs 실제 레이블)의 일치도. 1=완벽 일치, 0=무작위 수준(음수도 가능).
- 이상 탐지에서는 이상치로 표시된 샘플 중 실제 관심 클래스(예: 악성, 사기 거래)의 비율을 확인합니다. 이 가이드에서는 Isolation Forest가 이상치 중 악성 비율 66.7%로 가장 높았습니다(전체 악성 비율 37.3%). 다만 이상치의 약 1/3은 여전히 양성 종양이므로, 이상 탐지 결과를 곧바로 악성 판정으로 쓰면 안 됩니다.

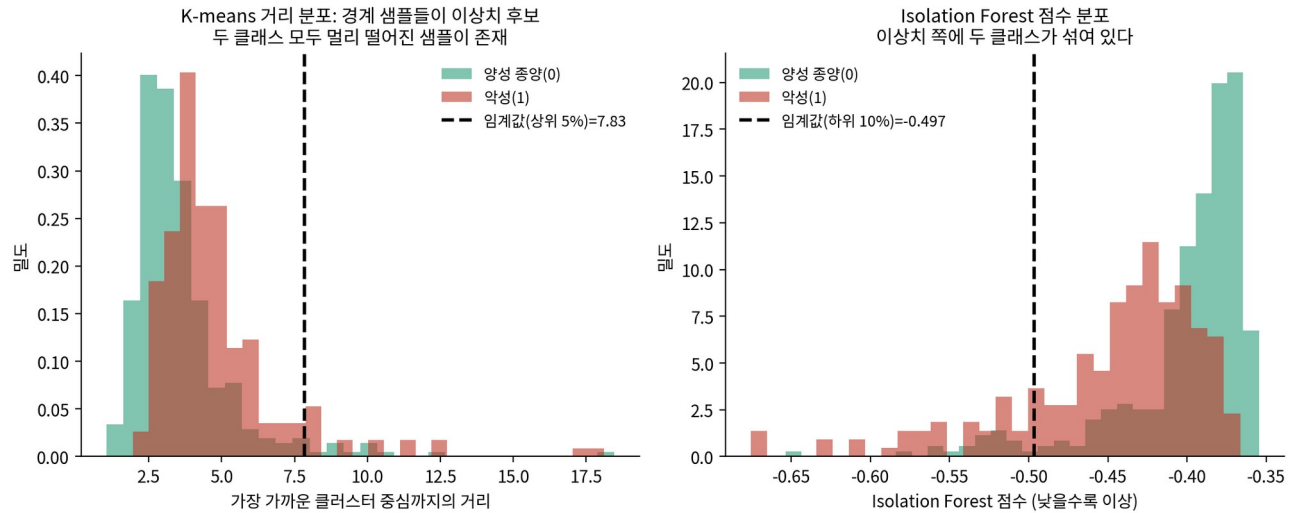


그림 5. 이상 탐지 점수 분포. K-means 거리(왼쪽): 양성 종양, 악성 모두 멀리 떨어진 샘플이 존재해, '이상치 = 특정 클래스'가 아님을 보여준다. Isolation Forest 점수(오른쪽): 임계값 왼쪽(이상치)에 두 클래스가 섞여 있다. 이상치가 반드시 악성인 것은 아니며, 레이블 있는 분류와 다른 질문에 답한다.

이상 탐지는 분류와 다른 질문에 답합니다

이상 탐지: '이 샘플은 다른 샘플들과 통계적으로 다른가?'

분류: '이 샘플은 어느 클래스에 속하는가?'

악성 종양이 이상치일 수도, 정상 범주 안에 있을 수도 있습니다. 이상치 = 악성이라고 단순화하면 위험합니다.

이상 탐지가 가장 효과적인 상황: 레이블이 없거나, 이상 사례가 매우 드물어 분류 모델 학습이 어려울 때.

4-3. 상황별 방법 선택 가이드

상황	추천 방법	이유
그룹 수가 대략 알려져 있다	K-means	단순하고 빠름, 해석 쉬움
그룹 수를 모르고 이상치가 있다	DBSCAN	k 불필요, 노이즈 자동 분류
전역적 이상치 탐지	Isolation Forest	빠르고 확장성 좋음
지역 밀도 기반 이상치	LOF	밀도 다른 클러스터 혼재 시 유리
클러스터링 전 권장	PCA로 차원 축소	노이즈 제거, 시각화, 계산 효율

이 가이드는 K-means(클러스터 수 선택, 한계), DBSCAN(밀도 기반, 노이즈 탐지), Isolation Forest와 LOF(이상 탐지 전용), 그리고 비지도학습 평가의 어려움을 다루었습니다. Gaussian Mixture Model(GMM), 계층적 클러스터링, One-Class SVM 등은 이 가이드의 범위를 벗어나며 필요에 따라 추가로 탐색할 수 있습니다.