머신러닝 기반의 이상탐지
Machine Learning-Based Anomaly Detection

ESM3081 데이터과학을위한프로그래밍

강석호 | 성균관대학교 산업공학과
s.kang@skku.edu

이상(Anomaly)

- **이상(Anomaly) 데이터**
  - 일반적인 데이터의 분포로부터 크게 벗어나는 데이터 값 또는 인스턴스를 의미
  - 일반적으로 이상 데이터는 비교적 희소하게 발생함
    - ✓ 예시: 제조 공정에서 불량 제품의 비율, 건강검진 시 건강 이상자의 비율
  - 이상의 원인에 따라 이상 인스턴스가 중요할 수도 있고, 무의미할 수도 있음. 맥락에 따른 판단 필요
    - ✓ 예시: 키가 2.3m인 사람 vs 200m인 사람, 혈압이 200mmHg인 환자 vs 1mmHg인 환자
    - ✓ 전자는 우리가 관심을 가지는 이상 케이스이나, 후자는 데이터 수집 오류의 가능성이 큼

- **이상(Anomaly) 관련 주요 개념**
  - **동의어/유의어:** Abnormal, Novelty, Outlier
  - **관련어:** Noise

이상(Anomaly)과 노이즈(Noise)의 차이

- 노이즈(Noise)
  - 노이즈는 잘못된 값 또는 그를 포함하는 인스턴스를 의미.
  - 그러나, 노이즈가 반드시 이상한 값이나 그를 포함하는 인스턴스이지는 않음.
  - 노이즈는 인스턴스가 가지는 특성을 왜곡할 수 있음.
  - 노이즈는 일반적으로 데이터의 활용 전에 미리 제거되어야 함.

- 이상(Anomaly)
  - 일반적인 데이터의 분포로부터 크게 벗어나는 데이터 값 또는 인스턴스를 의미.
  - 이상 데이터는 정상 데이터가 생성되는 매커니즘을 위반.
  - 노이즈와 이상은 관련은 있지만 분명히 별개의 개념임.
  - 이상의 원인이 노이즈가 아닌 경우 주로 우리가 관심을 가지는 대상이 됨.

이상(Anomaly)과 노이즈(Noise)의 차이

이상과 노이즈 여부 판단 예시

인스턴스 ID	키 실제값 (Unknown)	키 관측값
1	155cm	180cm
2	235cm	235cm
3	171cm	171cm
4	180cm	1.8cm
5	175cm	174cm
…	…	…

* 도메인 지식에 따른 판단 필요

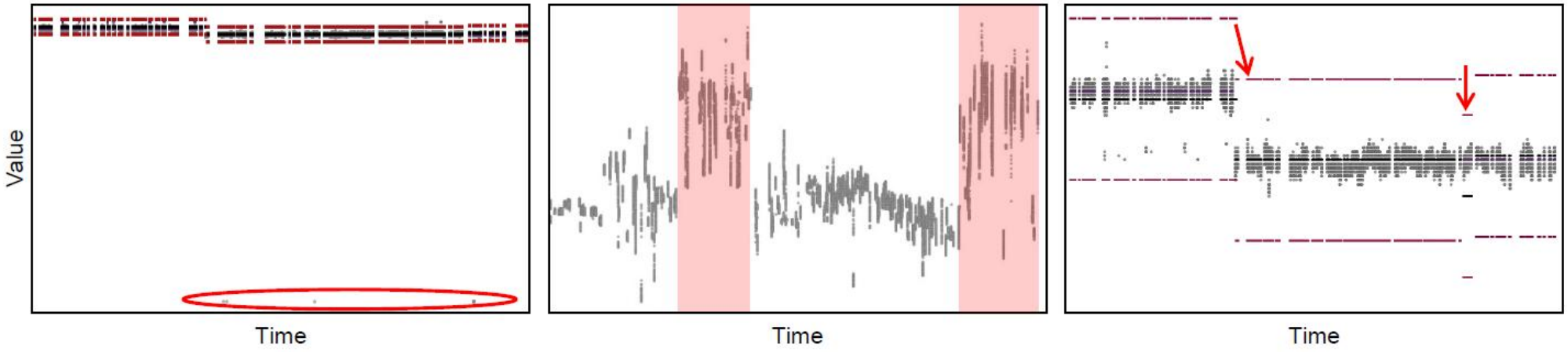
← 측정 오차가 크나, 관측값이 이상하지는 않음 (노이즈)

← 측정 오차는 없으나, 관측값이 이례적으로 큼 (이상)

← 관측값이 상식을 크게 벗어남 (이상이면서 노이즈)

이상(Anomaly)의 종류

- **단변량 이상 (Univariate Anomaly):** 하나의 변수에서 기존의 분포와 다른 새로운 값이 관측됨
  - 많은 현실 문제에서, 이상은 주로 특정 변수에서 단변량으로 발생함
  - 이상 탐지가 상대적으로 쉬움 (규칙 기반 탐지만으로도 성공적으로 탐지 가능)
  - **예시: 제조 공정에서 시간의 흐름에 따른 점진적/급진적 데이터 특성 변화**
    - ✓ **이상치(Outliers)** – 외부 충격, 불량 발생, 작업자 오류 (특히, 수기 데이터) 등
    - ✓ **점진적 데이터 특성 변화** – 설비 노후화, Calibration, 점진적 내/외부 환경 변화 등
    - ✓ **급진적 데이터 특성 변화** – 설비 고장, 제품/설비 조건 변경, 정비/유지보수 등



(a) Outliers (b) Seasonality (c) Process Drift

이상(Anomaly)의 종류

- **단변량 이상 (Univariate Anomaly):** 하나의 변수에서 기존의 분포와 다른 새로운 값이 관측됨
  - **단변량 이상 판별 규칙 예시**

과거 데이터	**100**, 107, 110, 112, 118, 122, 124, **130, 132**, 136, 140, 144, 148, 149, **149, 151**, 164, 168, 172, 176, 180, **184, 200**, 205, 219, 225, 235, 245, 255, **400**

✓ **Inter Quartile Range (IQR; 사분위범위):** Q3 – Q1 (과거 데이터에서 상위 25%값에서 하위 25%값을 뺀 값)
IQR = Q3 – Q1 = 200 – 130 = 70

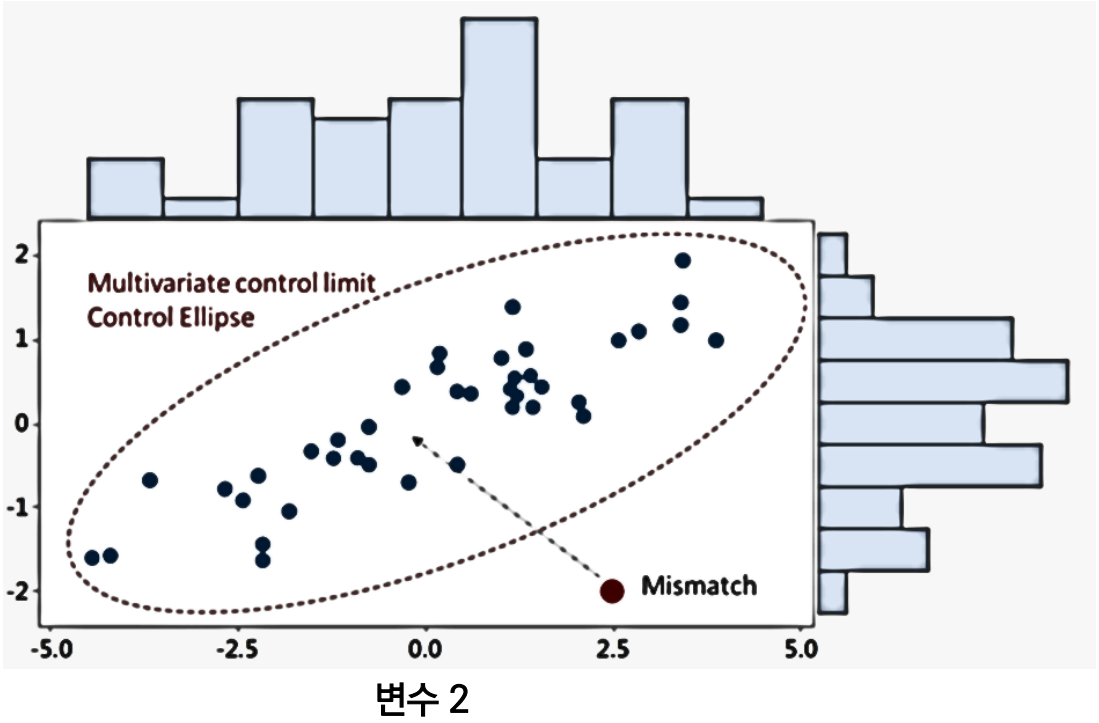
✓ **이상치:** Q1 – (1.5 x IQR) 보다 작은 값 또는 Q3 + (1.5 x IQR) 보다 큰 값
130 – 1.5 x 70 = 130 – 105 = 25 / 200 + 1.5 x 70 = 200 + 105 = 305

✓ **극단이상치:** Q1 – (3.0 x IQR) 보다 작은 값 또는 Q3 + (3.0 x IQR) 보다 큰 값
130 – 3.0 x 70 = 130 – 210 = –80 / 200 + 3.0 x 70 = 200 + 210 = 410

* 위 규칙은 이상치 판별의 절대적인 기준이 아니며, 이상치 일수도 있다는 신호로 판단하는 것이 타당함

이상(Anomaly)의 종류

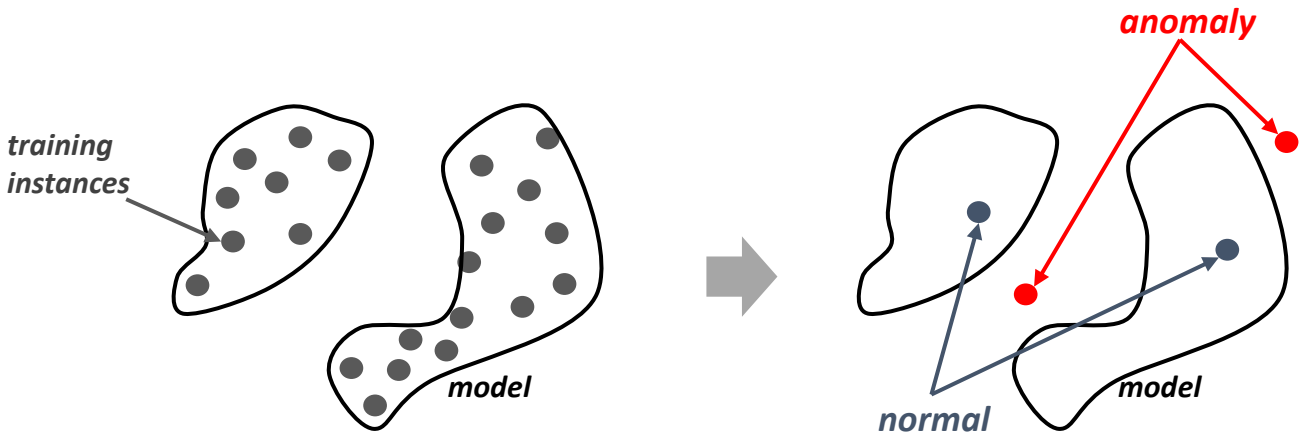
- **다변량 이상 (Multivariate Anomaly):** 여러 변수 간 관계를 볼 때, 기존의 관계 특성을 벗어나는 형태의 이상
  - 다변량 이상은 각 변수 별 단변량 이상으로는 판단되지 않을 수도 있음
  - 이상탐지가 상대적으로 어려움 (머신러닝 기반 이상탐지의 주요 탐지 목표)
  - **이변량 이상 예시**



이상탐지(Anomaly Detection)

- **주어진 데이터로부터 학습을 통해 대세 경향이나 분포를 따르지 않는 이상 인스턴스를 탐지**
  - **일반적으로, "이상탐지"는 비지도학습 기반의 단일범주분류(One-Class Classification)를 의미**
  - 학습 데이터가 대부분 정상 인스턴스들로 구성되며, 이상 인스턴스가 없거나 매우 적은 상황에 활용
    - ✓ **만약 학습 데이터가 충분한 수의 이상 인스턴스를 포함하고 있다면? → 지도학습 기반의 분류 접근 활용**
  - **[학습 단계]** 정상 인스턴스로만 구성된 학습 데이터셋을 이용하여 이상탐지 모델을 학습
  - **[적용 단계]** 이상탐지 모델을 적용하여, 미래의 쿼리 인스턴스에 대한 이상 여부 판단.
    - ✓ 모델이 인스턴스를 잘 설명하는 경우 → 정상으로 판단
    - ✓ 모델이 인스턴스를 잘 설명하지 못하는 경우 → 이상으로 판단, 추가 분석 수행

이상탐지 모델 학습 단계 **이상탐지 모델 적용 단계**

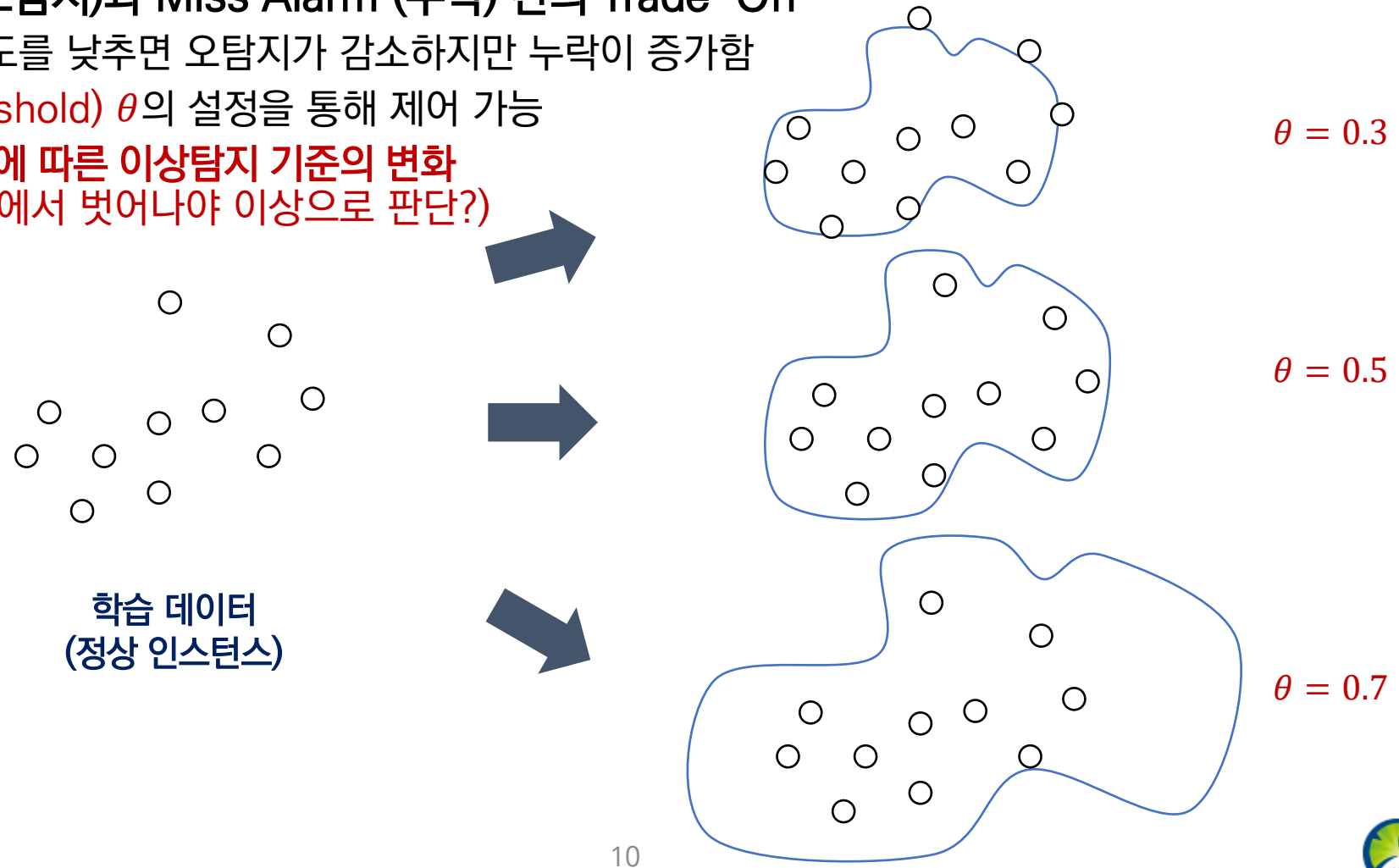


이상탐지(Anomaly Detection)

- **이상탐지 모델의 활용**
  - 새로운 쿼리 인스턴스가 주어졌을 때, 이상탐지 모델은 이상점수를 산출
  - 이상점수가 클수록 쿼리 인스턴스가 기존에 알려진 정상 인스턴스들의 경향이나 분포와 다름을 의미
  - 이상점수가 특정 임계값(Threshold) θ보다 크면 이상으로 판정

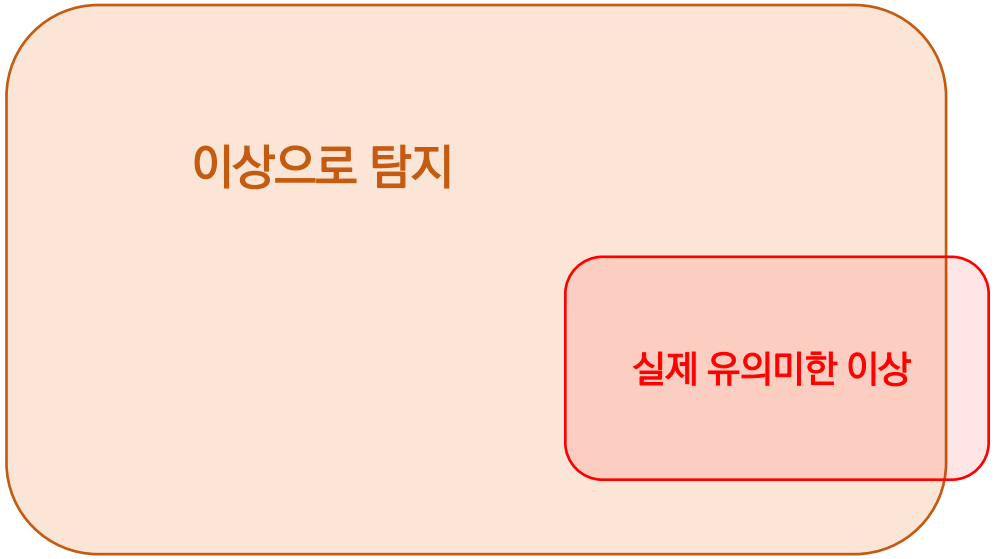
이상탐지(Anomaly Detection)

- False Alarm (오탐지)과 Miss Alarm (누락) 간의 Trade-Off
  - 모델의 민감도를 낮추면 오탐지가 감소하지만 누락이 증가함
  - 임계값(Threshold) θ의 설정을 통해 제어 가능
  - **예시: 임계값에 따른 이상탐지 기준의 변화**
    (얼마나 정상에서 벗어나야 이상으로 판단?)



이상탐지의 의미

- **이상으로 탐지된 모든 인스턴스가 실제로 유의미한 이상은 아닐 수 있음**
  - 이상탐지 모델은 기본적으로 기존의 학습 데이터 분포에 벗어나는 인스턴스를 이상으로 탐지
  - 단순히 어떤 인스턴스가 주어진 학습 데이터 분포에 벗어나는 것이 도메인 측면에서 항상 유의미한 이상을 의미하지는 않음.
    - ✓ False Alarm (오탐지)의 문제가 빈번하게 발생
  - **도메인 지식에 기반한 추가적인 분석을** 통한 의미 도출 필요.



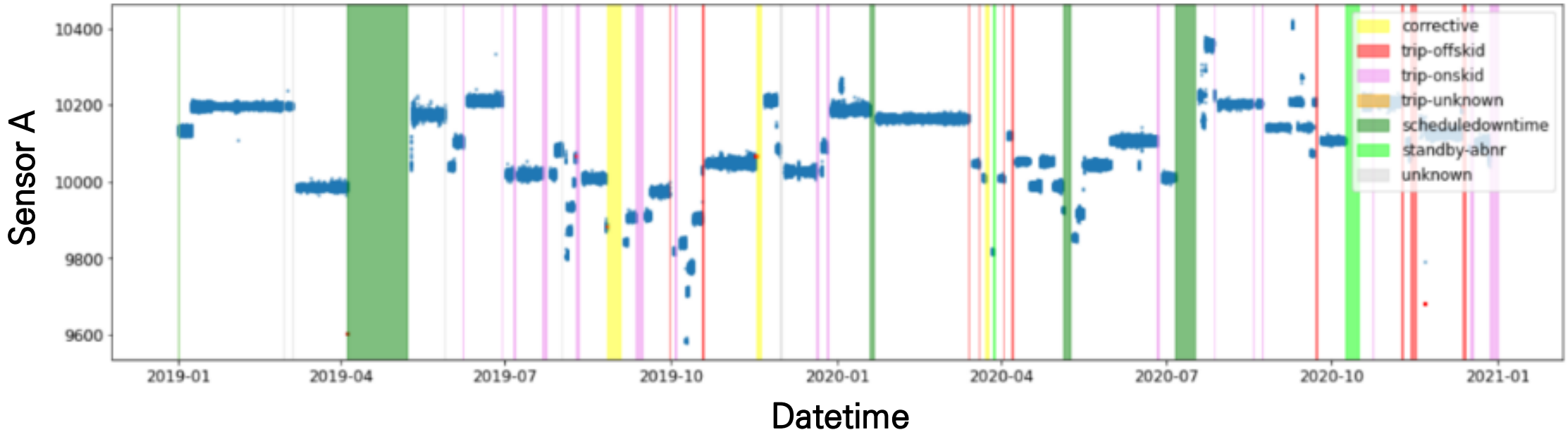
이상탐지의 의미

- 일반적으로 이상(Anomaly)의 판단을 위한 절대적인 기준은 없음
  - 도메인 지식에 기반한 이상에 대한 이해가 좋은 이상탐지 결과로 이어질 수 있음
  - 예시: 아래 이미지들 중 어떤 이미지가 나머지와 특성이 다른가요?

???

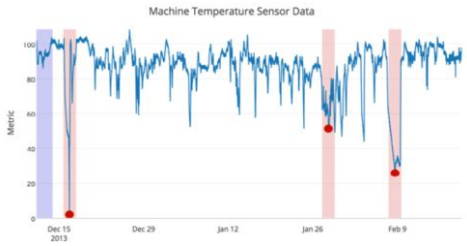
이상탐지의 의미

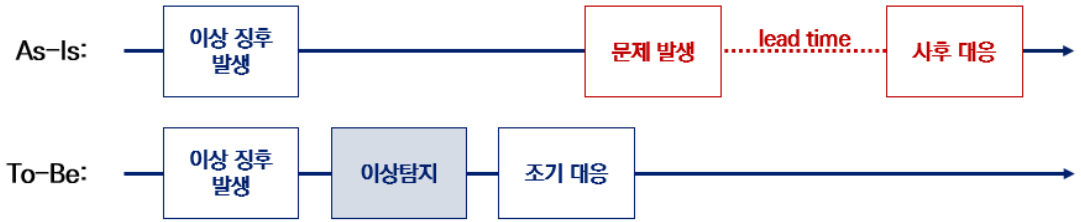
- 일반적으로 이상(Anomaly)의 판단을 위한 절대적인 기준은 없음
  - 도메인 지식에 기반한 이상에 대한 이해가 좋은 이상탐지 결과로 이어질 수 있음
  - 예시: 아래 시계열 데이터에서 어느 시점의 관측값들이 이상인가요?
    - ✓ Local vs Global Anomaly



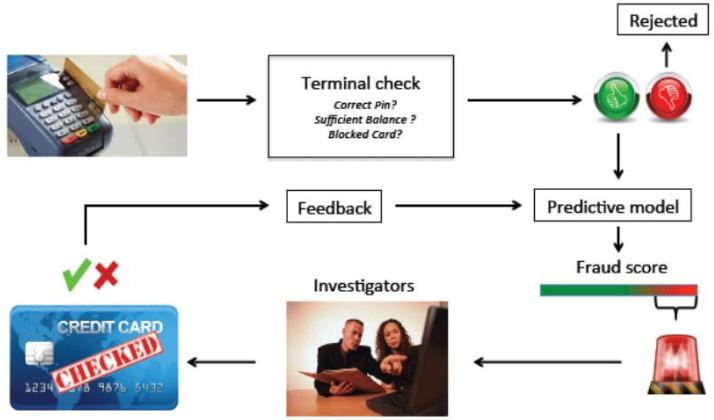
이상탐지 적용 사례

- 설비 모니터링 및 예지보전

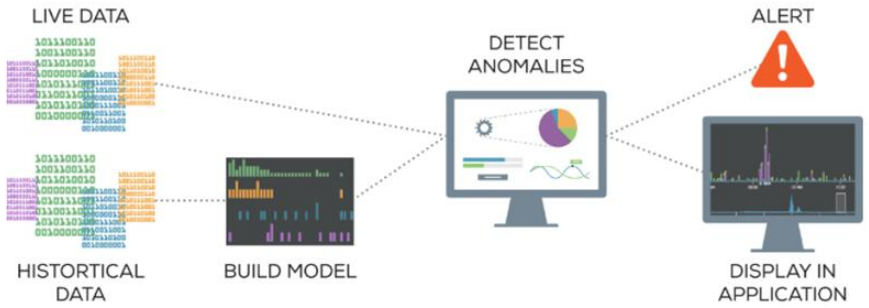




- 카드 이용 부정 탐지



- 시스템 로그 기반 이상 탐지



- 불량 혐의 제품 판별
- 의료 진단
- 특이 고객 판별
- …

이상탐지 고려사항

- **학습 데이터가 발생 가능한 정상 케이스를 충분히 포함하고 있음을 간주**
  - 학습 데이터가 설명하지 못하는 정상 인스턴스는 이상탐지 모델이 이상으로 탐지하게 됨

- **현실 문제에서, 정상과 이상 간의 경계가 모호한 경우가 많음**
  - 어떤 변수에 대해서 정상과 얼마나 달라야 이상인지에 대한 기준은 주관적
    - ✓ 예시: 사람의 키가 몇 cm까지 정상이고, 몇 cm부터 이상인가?
      사람의 몸무게는 몇 kg까지 정상이고, 몇 kg부터 이상인가?
  - 개별 인스턴스의 이상 여부에 대한 이진 판단보다는,
    이상의 정도를 나타내는 이상점수(Anomaly Score)를 활용 가능

- **노이즈는 이상탐지를 왜곡하여 성능을 저하시킬 수 있음**
  - 노이즈는 정상과 이상 간의 경계를 더욱 모호하게 함 (이상과 노이즈가 구분이 안됨)
  - 노이즈를 식별하고 사전에 제거하는 것은 매우 어려움

이상탐지 고려사항

- **현실 문제에서 이상탐지 모델의 성능을 정량적으로 평가하는 것은 매우 어려움**
  - 특히, 이상 데이터의 부재 상황에서 이상탐지 모델을 평가하는 것은 불가능
  - 인위적으로 이상 데이터를 생성하거나, 이상탐지 모델이 도출하는 결과에 대한 전문가의 판단에 따른 검사 필요

- **이상으로 탐지된 인스턴스가 왜 이상인지에 대한 설명을 통해 이상탐지 결과를 정당화할 수 있음**
  - 도메인 지식에 따른 사후해석이 매우 중요하며, 최근 XAI기법을 적용하는 시도들이 이루어지고 있음.

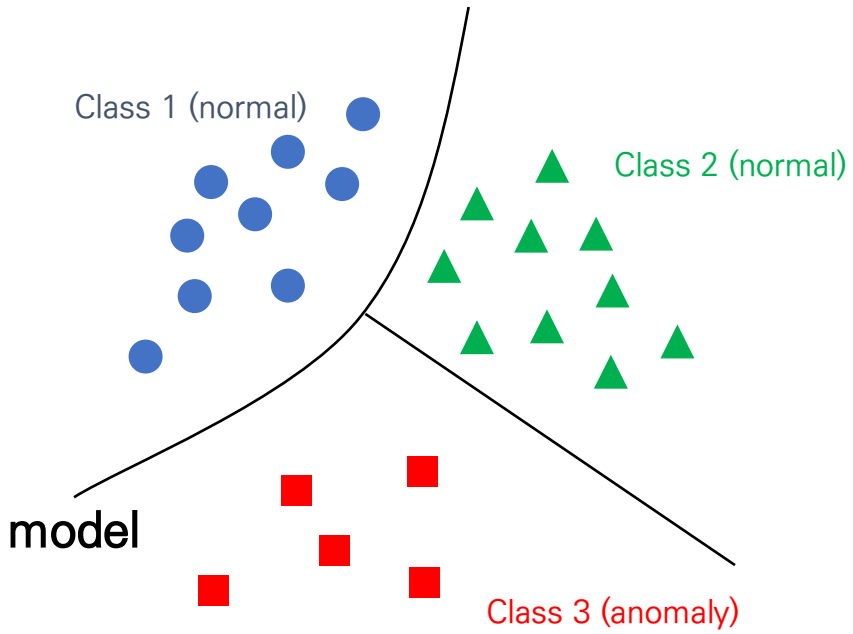
머신러닝 기반 이상탐지

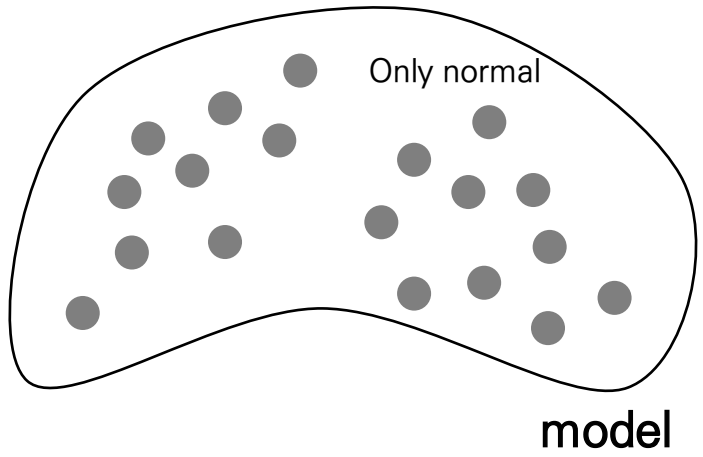
- **지도학습 접근 (Supervised Anomaly Detection) → 범주 불균형을 고려한 분류(Classification)**
  - 학습 데이터 내 이상 인스턴스가 별도의 범주로 존재함
  - 일반적으로 이상은 정상에 비해 희소하게 발생하며,
    따라서 이상 인스턴스의 수가 정상 대비 매우 적은 범주 불균형(Class Imbalance) 문제가 발생
  - 학습 데이터에 존재하지 않는 유형의 이상 인스턴스에 대해서 성능이 저하될 수 있음

- **비지도학습 접근 (Unsupervised Anomaly Detection) → 단일범주분류(One-Class Classification)**
  - **일반적으로 머신러닝 기반의 이상탐지는 이 접근을 의미함**
  - 정상 인스턴스로만 구성된 학습 데이터로부터 이상탐지 모델을 학습
  - 학습된 이상탐지 모델은 새로운 쿼리 인스턴스를 정상 또는 이상(Anomaly)으로 판정 (또는 이상점수를 제공)
  - 주어진 쿼리 인스턴스가 이상으로 판단됨은, 이상탐지 모델이 해당 인스턴스를 설명하지 못함을 의미

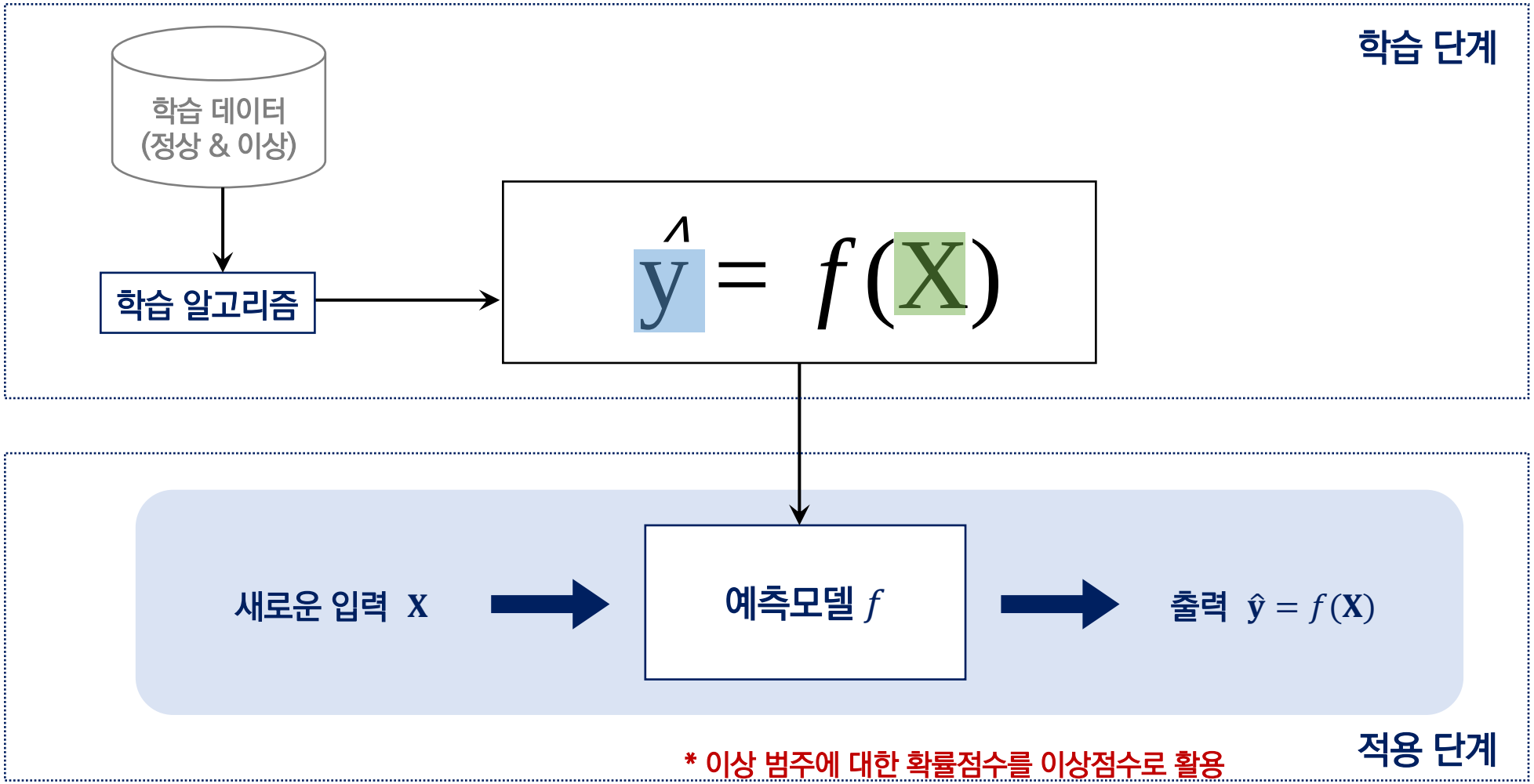
머신러닝 기반 이상탐지

지도학습 접근 vs 비지도학습 접근





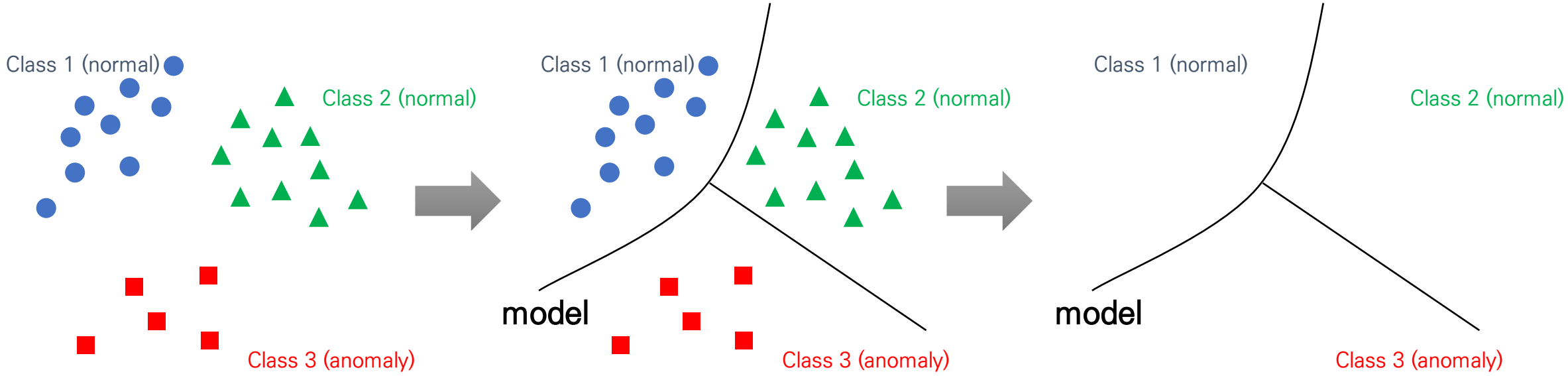
머신러닝 기반 이상탐지 – 지도학습 접근



* 이상 범주에 대한 확률점수를 이상점수로 활용
* 이상점수가 특정 임계값 θ보다 크면 이상으로 판정

머신러닝 기반 이상탐지 – 지도학습 접근

- **학습 단계:** 정상과 이상 인스턴스를 모두 포함하는 학습 데이터로부터 범주 간 구분 경계를 형성하는 분류모델 구축



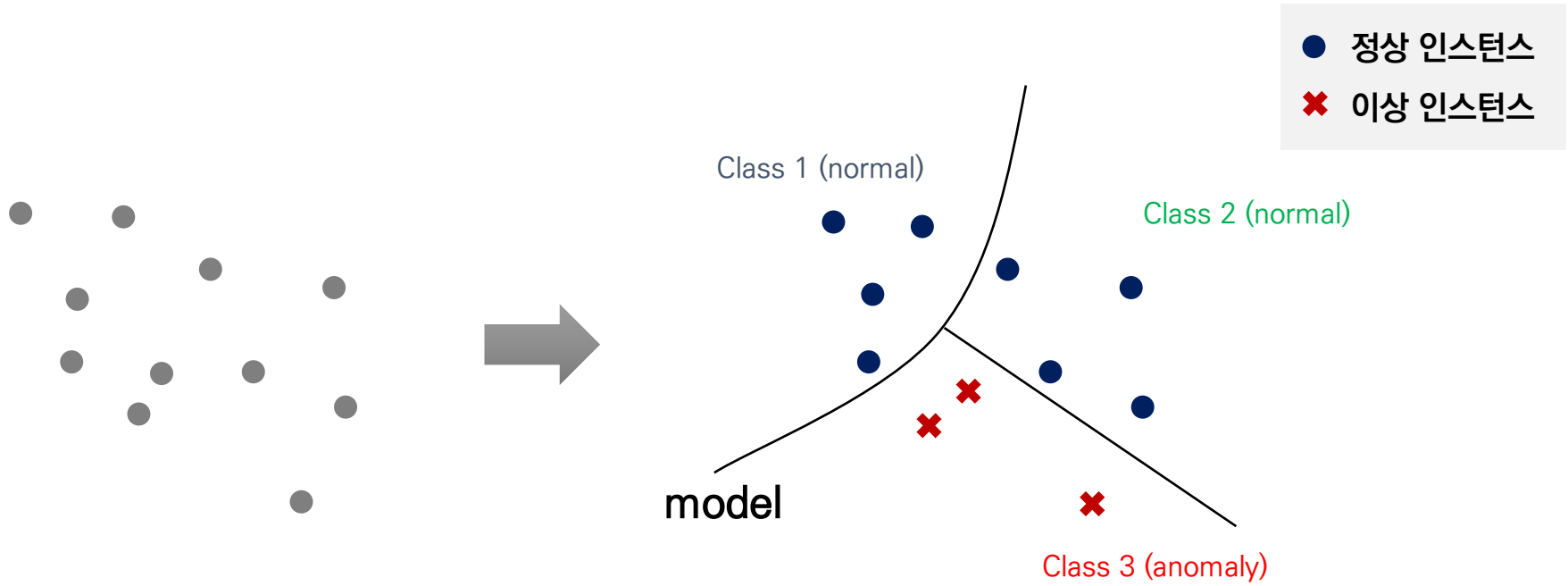
step 1 : 학습 데이터

step 2 : 범주 간 구분 경계를 형성하는 분류모델 학습

step 3 : 학습된 분류모델

머신러닝 기반 이상탐지 – 지도학습 접근

- **적용 단계:** 분류모델에 의해 이상 범주 영역으로 구분되는 인스턴스를 이상(Anomaly)으로 판단



step 1 : 새로운 데이터

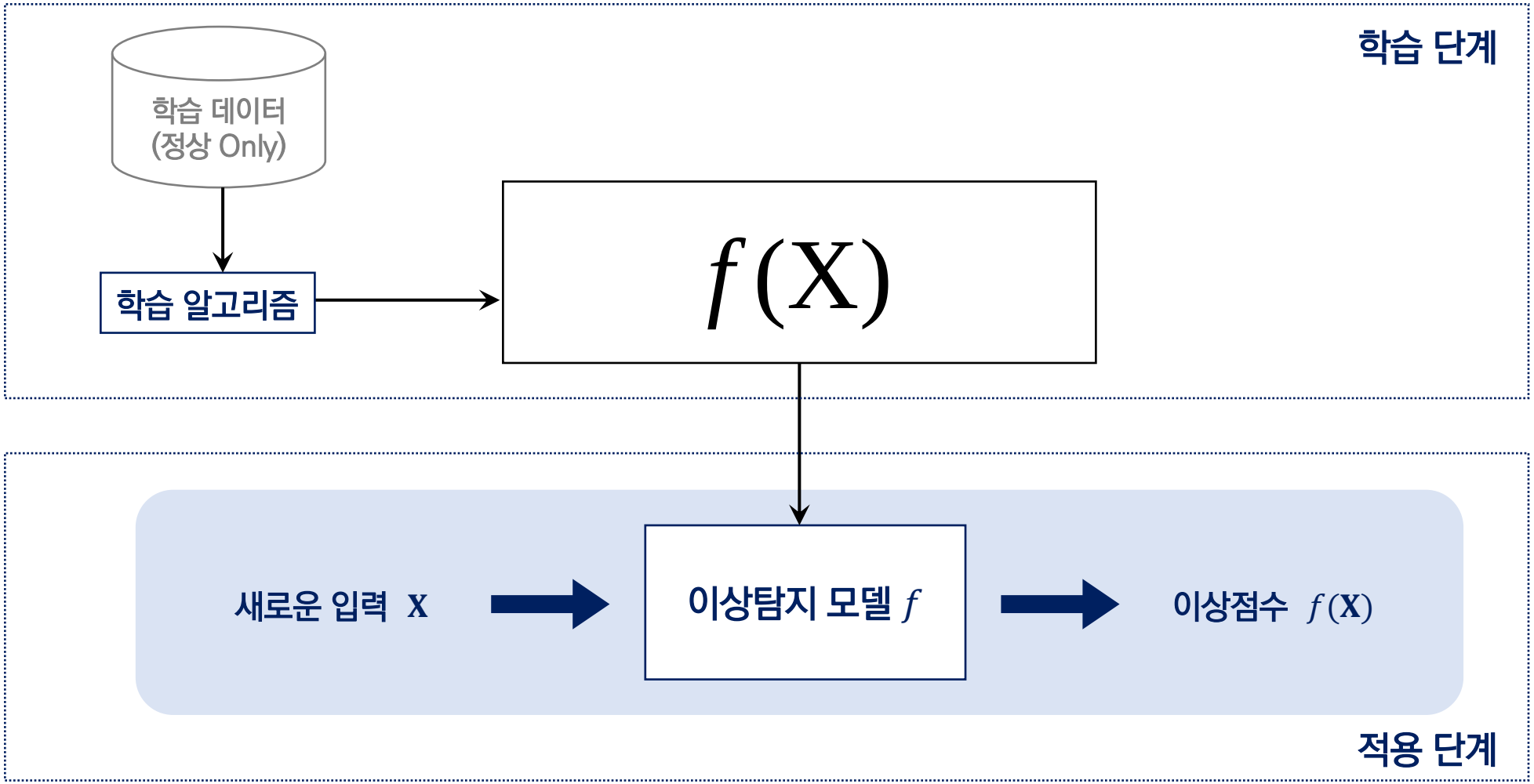
step 2 : 분류모델을 이용한 데이터 구분

머신러닝 기반 이상탐지 – 지도학습 접근

- 이상의 기준이 명확하고, 학습 데이터로 이상 인스턴스를 충분히 확보 가능하다면 최선의 접근임
- 일반적으로 이상은 정상 대비 드물게 발생하며, 따라서 이상 인스턴스 수는 정상 대비 매우 적음
- 지도학습의 분류 방법론들은 일반적으로 범주 간 균형을 간주하고 있으며,
  따라서 이상 인스턴스가 부족한 범주 불균형 상황에서 성능이 크게 저하될 수 있음

- 이상 인스턴스 수가 극단적으로 적은 상황에서는
  비지도학습 기반의 이상탐지를 통해서 더 나은 성능을 얻을 수도 있음

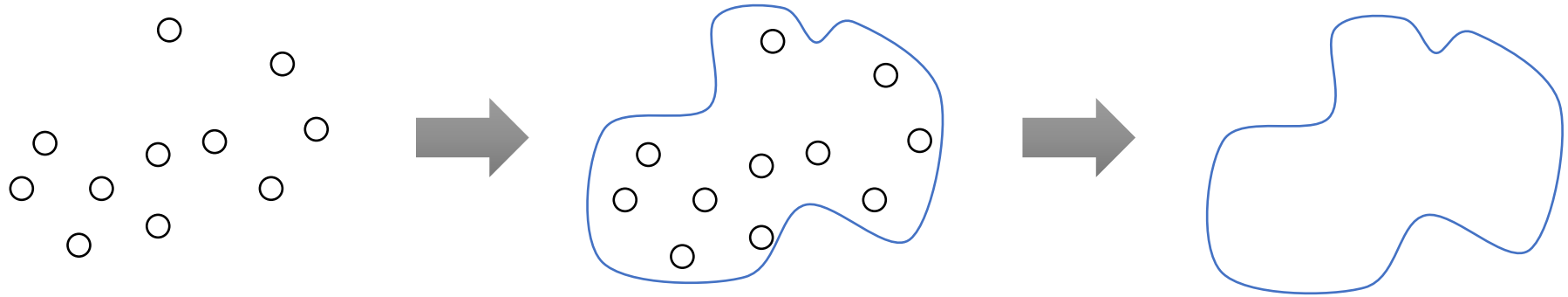
머신러닝 기반 이상탐지 – 비지도학습 접근



* 이상점수가 특정 임계값 θ보다 크면 이상으로 판정

머신러닝 기반 이상탐지 – 비지도학습 접근

- **학습 단계:** 정상 인스턴스로만 구성된 학습 데이터의 분포를 설명하는 이상탐지 모델 구축.



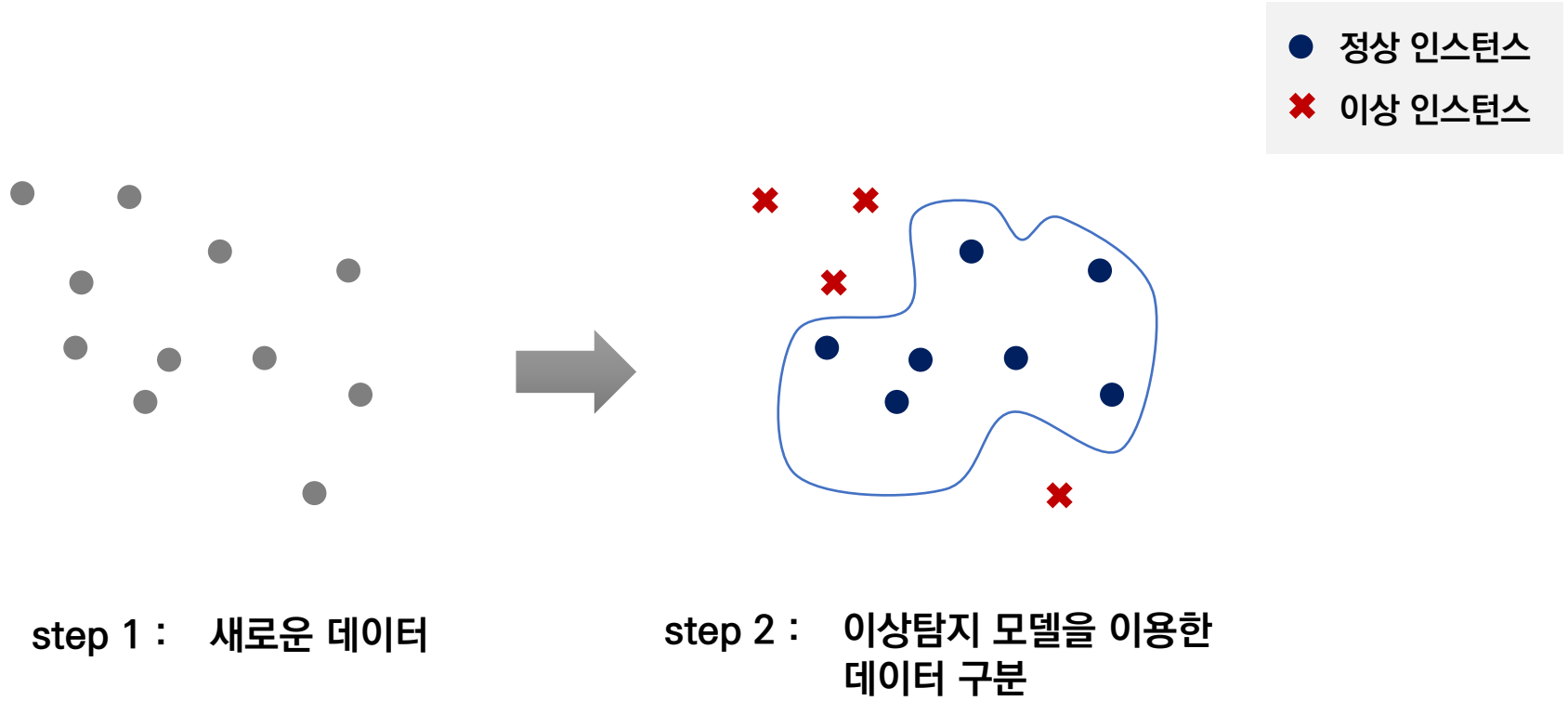
step 1 : 학습 데이터 (정상 인스턴스)

step 2 : 학습 데이터 분포를 설명하는 이상탐지 모델 학습

step 3 : 학습된 이상탐지 모델

머신러닝 기반 이상탐지 – 비지도학습 접근

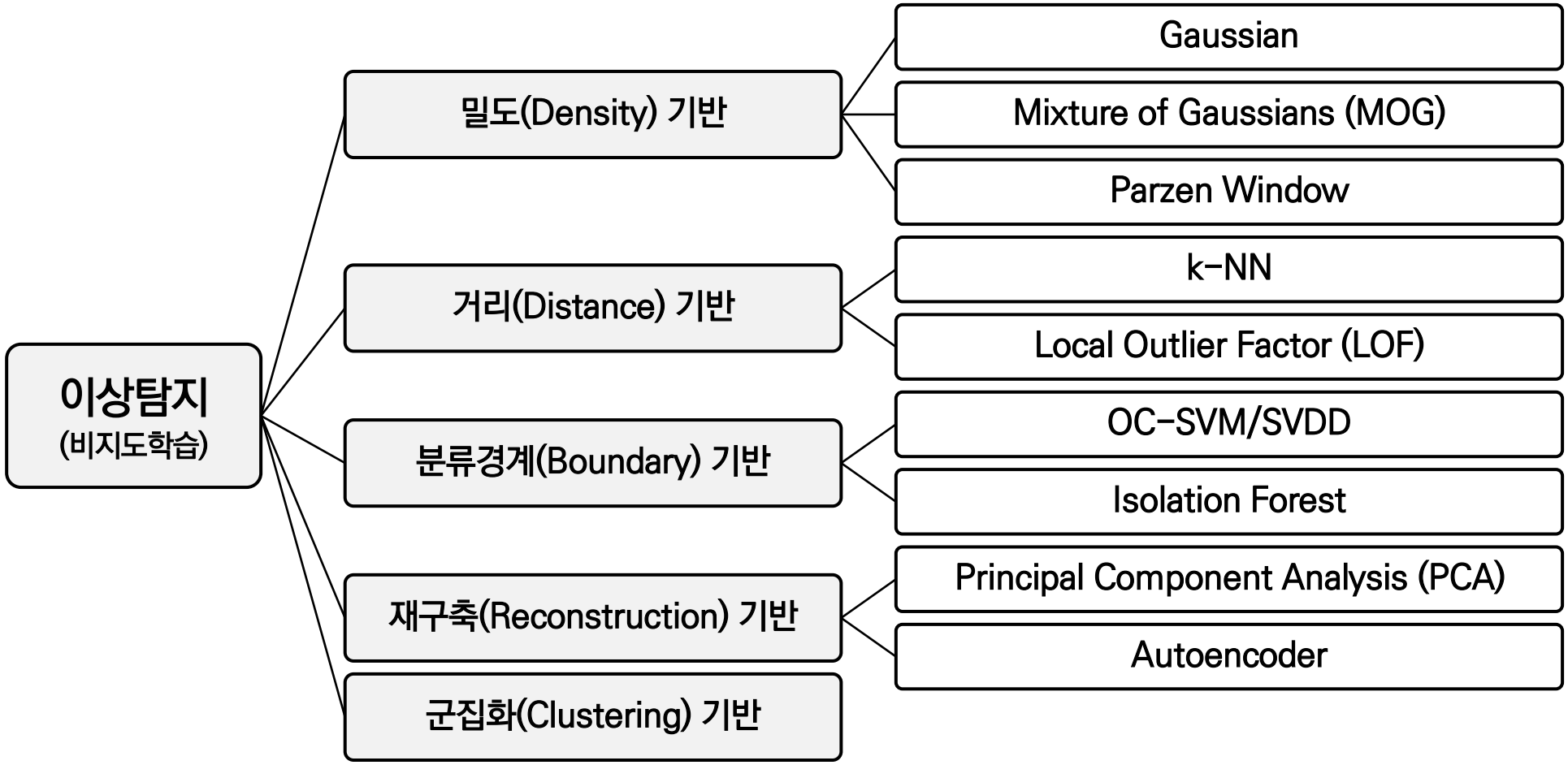
- **적용 단계:** 이상탐지 모델이 설명하는 영역 바깥에 있는 인스턴스를 이상(Anomaly)으로 판단



머신러닝 기반 이상탐지 – 비지도학습 접근

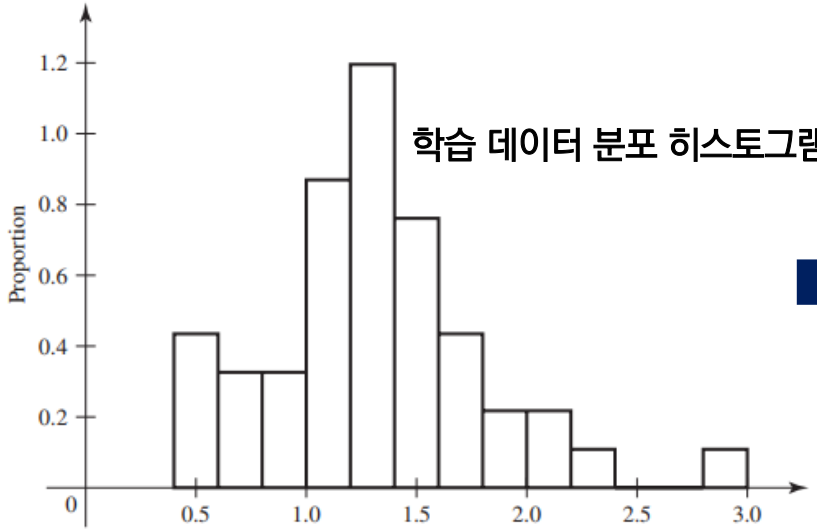
- 학습 데이터에 이상 인스턴스가 부재한 상황에서도 활용 가능
- 기존에 알려지지 않았던 이상 케이스를 식별하여 탐지하는데 효과적임
- 기존에 알려진 정상 데이터 분포에 벗어나는 모든 인스턴스를 이상으로 식별하며, 이 과정에서 False Alarm (오탐지)의 문제가 빈번하게 발생
- 정량적인 성능평가가 어려우며, 따라서 모델의 하이퍼파라미터 최적화가 어려움

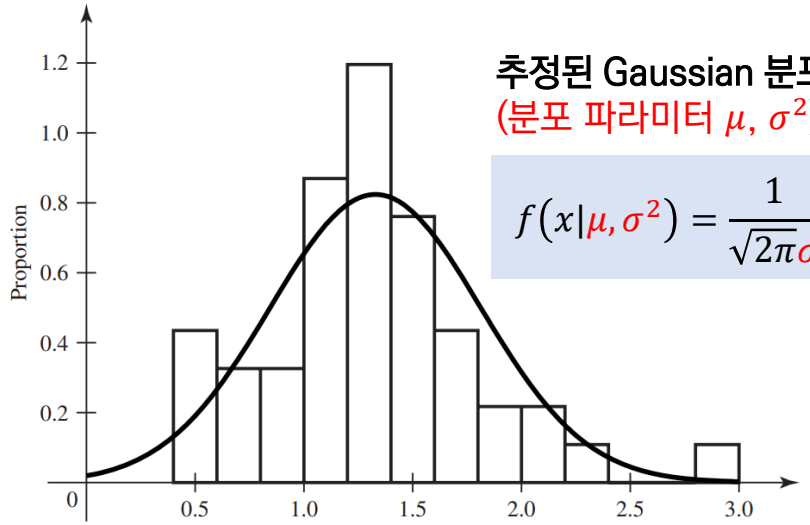
주요(전통적) 방법론의 구분



밀도(Density) 기반 이상탐지

- **쿼리 인스턴스가 학습 데이터의 분포 상에서 확률밀도가 낮은 영역에 속하는 경우 이상으로 판단**
  - 학습 데이터(정상 인스턴스로 구성)가 특정 확률 분포를 따름을 가정
    - ✓ 가정한 확률 분포가 데이터에 적합할수록 좋은 성능 (적합하지 않은 경우 나쁜 성능)
    - ✓ 확률 분포에 대한 사전 정보가 주어진 경우 효율적
  - **학습 단계:** 학습 데이터의 분포를 설명하는 확률 분포에 대한 파라미터를 추정하여 확률밀도함수(Probability Density Function) f를 도출
  - **적용 단계:** 주어진 쿼리 인스턴스 $\mathbf{x}$에 대해 확률밀도 값 $f(\mathbf{x})$가 작을수록 높은 이상점수 부여
  - 예시: 밀도 기반 단변량 이상탐지 (Gaussian 분포 가정)





scikit-learn implementation:
sklearn.covariance.EllipticEnvelope

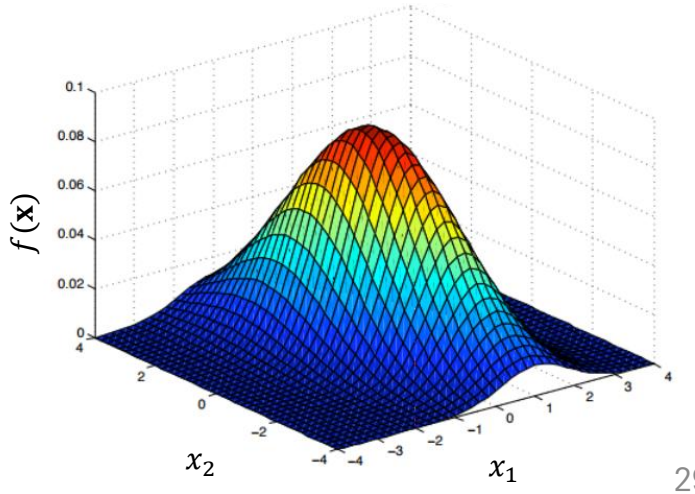
밀도(Density) 기반 이상탐지

- **Gaussian Density Estimation:** 학습 데이터가 Gaussian(정규) 분포를 따름을 가정하고 분포 파라미터를 추정하는 방법
  - Gaussian 분포의 확률밀도함수

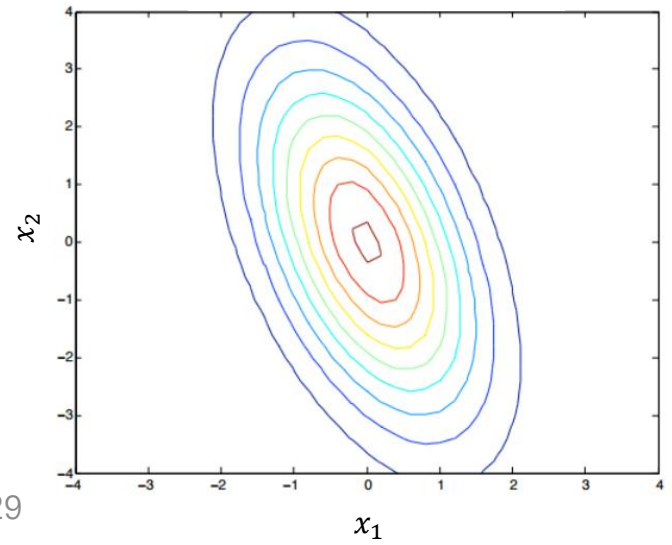
$$f(\mathbf{x}) = N(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$$

  - 최대우도추정(Maximum Likelihood Estimation)을 통해 학습 데이터를 잘 설명하는 분포 파라미터 $\boldsymbol{\mu}$와 $\boldsymbol{\Sigma}$의 값을 추정 가능
  - 학습 데이터의 확률분포에 대한 매우 강한 가정(Unimodal Gaussian)을 가지고 있으며, 복잡한 데이터 분포를 설명하는데 적합하지 않음

Surface Plot of $f(\mathbf{x})$



Contour Plot of $f(\mathbf{x})$



scikit-learn implementation:
sklearn.mixture.GaussianMixture

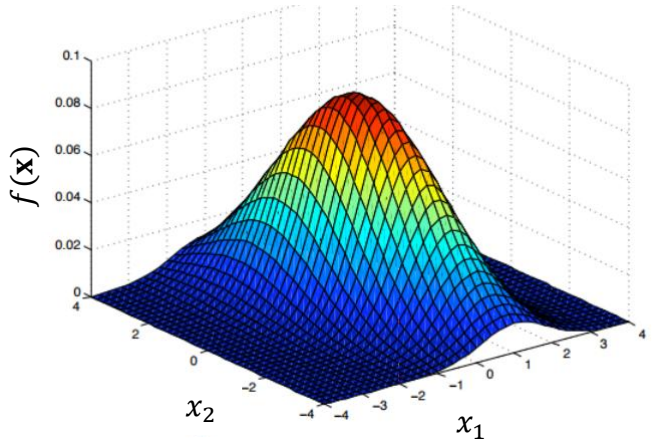
밀도(Density) 기반 이상탐지

- **Mixture of Gaussians (MOG):** 학습 데이터가 K개의 Gaussian(정규) 분포의 혼합 분포를 따름을 가정하고 분포 파라미터를 추정하는 방법
  - 혼합 분포의 확률밀도함수 (K개의 Gaussian 분포에 대한 확률밀도함수의 선형 조합)

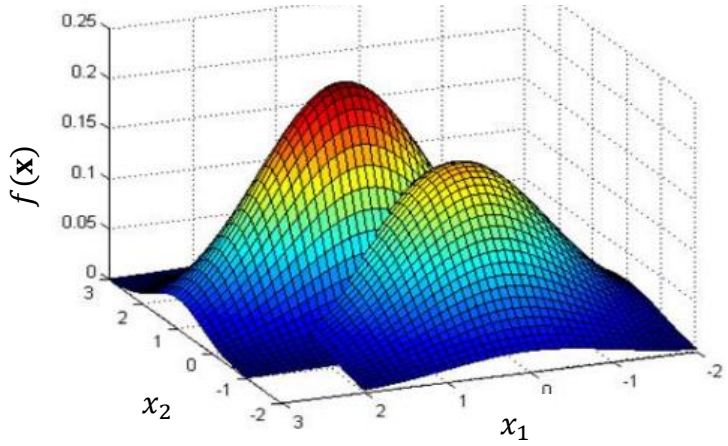
$$f(\mathbf{x}) = \sum_{k=1}^{K} w_k N(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  - Expectation-Maximization 알고리즘을 활용하여 파라미터 추정 가능
  - **중요 하이퍼파라미터: Gaussian 분포 개수 K (클수록 혼합 분포의 봉우리 개수가 늘어남)**
    - ✓ K=1이면 Gaussian Density Estimation과 동일
    - ✓ K가 클수록 더 복잡한 분포의 형태 표현이 가능해지나, 분포 파라미터의 추정에 많은 시간이 소요됨

One Gaussian



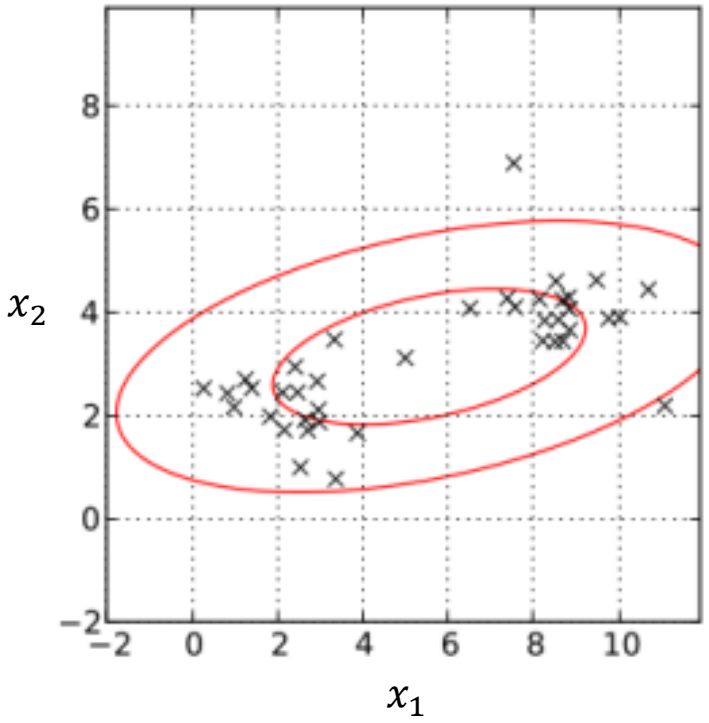
Mixture of Two Gaussians



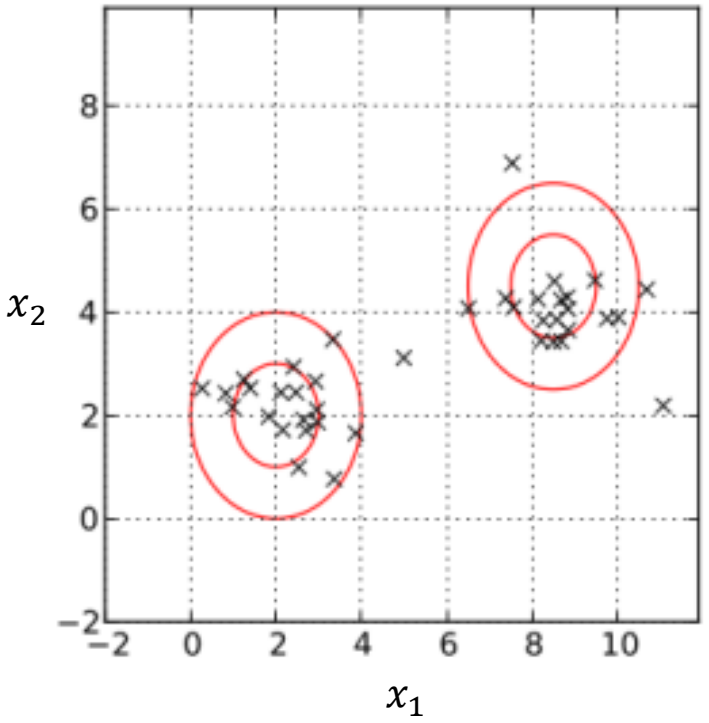
밀도(Density) 기반 이상탐지

예시: 2차원 학습 데이터에 대한 Gaussian과 MOG의 추정 확률밀도함수 비교

Contour Plot of $f(\mathbf{x})$ by
Gaussian Density Estimation



Contour Plot of $f(\mathbf{x})$ by
Mixture of Gaussians (K=2)



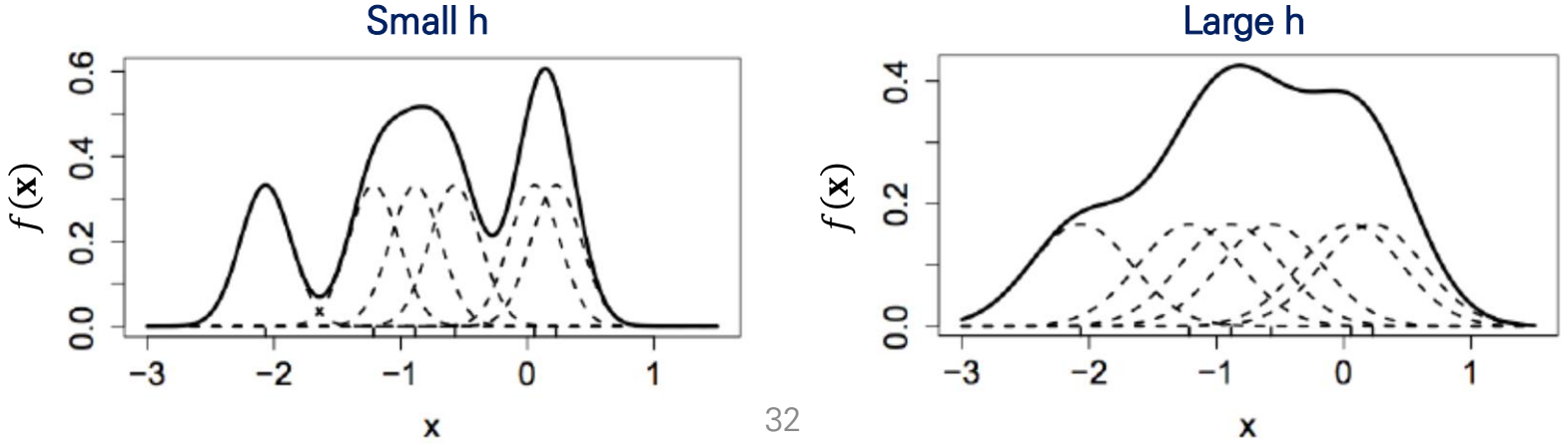
scikit-learn implementation:
sklearn.neighbors.KernelDensity

밀도(Density) 기반 이상탐지

- Parzen Window (a.k.a., Kernel Density Estimation): 학습 데이터에 대한 별도의 분포 가정 없이, 개별 데이터 인스턴스를 중심으로 하는 확률분포의 혼합 분포를 활용하는 방법
  - 혼합 분포의 확률밀도함수
    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$
    * k는 kernel 함수 (개별 데이터 인스턴스를 중심으로 하는 확률분포의 모양을 결정), d는 데이터의 차원

$$f(\mathbf{x}) = \frac{1}{n}\sum_{\mathbf{x}_i \in D} \frac{1}{h^d} k\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|}{h}\right)$$

  - 별도의 분포 파라미터 추정이 불필요하나, 쿼리 인스턴스에 대한 이상점수 계산 비용이 상대적으로 큼
  - **중요 하이퍼파라미터: bandwidth h (클수록/작을수록 혼합 분포의 봉우리 모양이 완만/뾰족해짐)**
    ✓ 분포에 대한 별도 가정 없이 높은 자유도로 복잡한 분포 형태를 표현할 수 있으나, $f(\mathbf{x})$의 형태가 h의 크기에 민감



거리(Distance) 기반 이상탐지

- **쿼리 인스턴스가 학습 데이터 인스턴스들과 상대적으로 멀리 떨어진 경우 이상으로 판단**
  - 특정 인스턴스와 그 이웃 인스턴스들 간 거리가 멀다면 데이터 공간 상 밀도가 낮은 영역에 위치함을 간주하여, 거리 기반으로 밀도를 표현할 수 있음
    - ✓ 정상 인스턴스의 밀도는 이웃 인스턴스들의 밀도와 비슷하거나 클 것임
    - ✓ 이상 인스턴스의 밀도는 이웃 인스턴스들의 밀도보다 크게 작을 것임
  - **학습 단계:** 없음
  - **적용 단계:** 주어진 쿼리 인스턴스 $\mathbf{x}$에 대해 학습 데이터셋 내 가까운 인스턴스들을 선택하고 이들과의 거리 관계를 바탕으로 이상점수 부여
  - 학습 데이터셋의 크기가 클수록 쿼리 인스턴스에 대한 이상점수 계산에 시간이 더 오래 걸림.
  - 주의: 인스턴스간 거리를 적절하게 정의하여 더 나은 이상탐지 결과를 얻을 수 있음
    - ✓ 변수 선택, 변수 스케일링, 거리지표의 적절한 선택이 중요함

거리(Distance) 기반 이상탐지

- **k-Nearest Neighbor (k-NN) 기반 이상탐지:** 쿼리 인스턴스의 학습 데이터셋 내 최근접 이웃 인스턴스들로부터의 거리를 활용하여 이상점수를 계산하는 방법
  - **이상점수 계산 예시 (다양한 방법들이 제안되어 옴)**

    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 쿼리 인스턴스 $\mathbf{x}$에 대한 학습 데이터셋 D 내 j번째 이웃 $N_j(\mathbf{x}) \in D$

    ✓ 쿼리 인스턴스 $\mathbf{x}$와 k번째 이웃 인스턴스 $N_k(\mathbf{x})$ 간 거리 (절대적 거리 기반 판단)

    $$f(\mathbf{x}) = d\big(\mathbf{x}, N_k(\mathbf{x})\big)$$

    ✓ 쿼리 인스턴스 $\mathbf{x}$와 k개의 최근접 이웃 간 평균 거리 (절대적 거리 기반 판단)

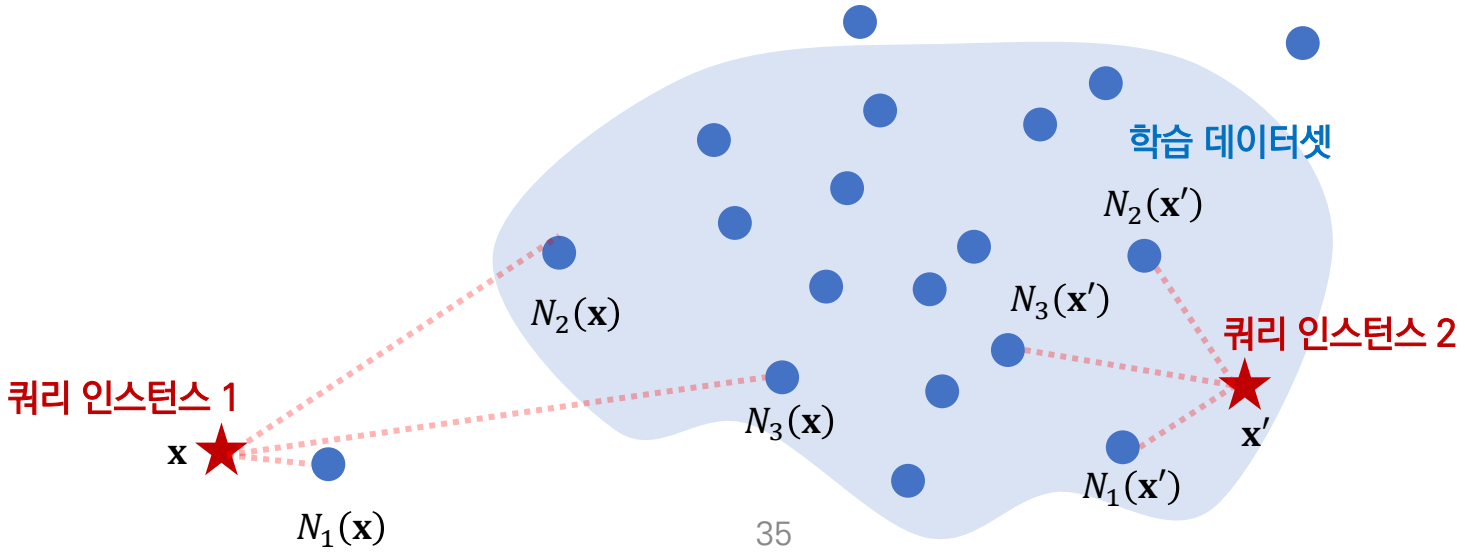
    $$f(\mathbf{x}) = \frac{1}{k}\sum_{j=1}^{k} d\left(\mathbf{x}, N_j(\mathbf{x})\right)$$

    ✓ 쿼리 인스턴스 $\mathbf{x}$와 k번째 이웃 인스턴스 $N_k(\mathbf{x})$ 간 거리를 $N_k(\mathbf{x})$의 k번째 이웃 인스턴스 $N_k\big(N_k(\mathbf{x})\big)$로부터의 거리로 정규화 (상대적 거리 기반 판단)

    $$f(\mathbf{x}) = d\big(\mathbf{x}, N_k(\mathbf{x})\big) / d\left(N_k(\mathbf{x}), N_k\big(N_k(\mathbf{x})\big)\right)$$

거리(Distance) 기반 이상탐지

- **k-Nearest Neighbor (k-NN) 기반 이상탐지:** 쿼리 인스턴스의 학습 데이터셋 내 최근접 이웃 인스턴스들로부터의 거리를 활용하여 이상점수를 계산하는 방법
  - **중요 하이퍼파라미터:** 쿼리 인스턴스의 이웃 개수 k (+ 인스턴스 간 거리의 기준)
    - ✓ 이상탐지 결과가 k의 설정에 크게 영향을 받음
    - ✓ 작을수록 노이즈/Outlier에 민감, 클수록 데이터의 지역적(Local) 특성을 무시
  - 예시: 두 쿼리 인스턴스 $\mathbf{x}$와 $\mathbf{x}'$ 의 최근접 이웃 인스턴스들과의 거리 관계
    - ✓ k=1일 때 $\mathbf{x}$의 이상점수가 $\mathbf{x}'$의 이상점수보다 낮게 나오나, $N_1(\mathbf{x})$은 학습 데이터셋의 Outlier로 판단됨
    - ✓ k가 매우 커지면 $\mathbf{x}$와 $\mathbf{x}'$의 이상점수의 차이가 거의 없어짐



scikit-learn implementation:
sklearn.neighbors.LocalOutlierFactor

거리(Distance) 기반 이상탐지

- Local Outlier Factor (LOF): 쿼리 인스턴스 주변의 국지적 밀도(Local Density)가 학습 데이터셋 내 최근접 이웃 인스턴스들의 국지적 밀도보다 상대적으로 낮으면 이상으로 판단하는 방법
  - **적용 단계에서 쿼리 인스턴스 $\mathbf{x}$에 대한 이상점수 계산 절차**

    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 쿼리 인스턴스 $\mathbf{x}$에 대한 학습 데이터셋 D 내 j번째 이웃 $N_j(\mathbf{x}) \in D$

    Step 1. 학습 데이터셋 D로부터 쿼리 인스턴스 $\mathbf{x}$의 k개의 최근접 이웃 $N_1(\mathbf{x}), \dots, N_k(\mathbf{x})$ 을 찾음

    Step 2. 쿼리 인스턴스 $\mathbf{x}$와 각 이웃 인스턴스 $N_j(\mathbf{x})$에 대한 Reachability Distance를 계산
    ($\mathbf{x}$와 $N_j(\mathbf{x})$ 간의 거리를 계산하나, 거리가 너무 작으면 $N_j(\mathbf{x})$의 k번째 이웃 $N_k(N_j(\mathbf{x}))$으로부터의 거리로 대체)

    $$\text{reach\_dist}_k(\mathbf{x}, N_j(\mathbf{x})) = \max\{d(N_j(\mathbf{x}), N_k(N_j(\mathbf{x}))), d(\mathbf{x}, N_j(\mathbf{x}))\}$$

    Step 3. 쿼리 인스턴스 $\mathbf{x}$와 k개의 이웃 간 Reachability Distance 평균의 역수로 Local Reachability Density 계산
    (쿼리 인스턴스가 최근접 이웃들과 가까울수록 큰 값을 가짐 – 쿼리 인스턴스 주변의 국지적 밀도를 의미)

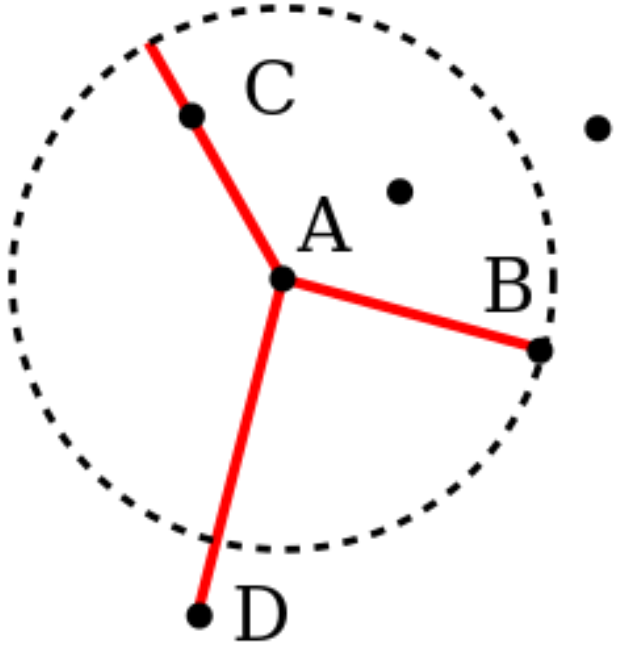
    $$\text{lrd}_k(\mathbf{x}) = \frac{k}{\sum_{j=1}^{k} \text{reach\_dist}_k(\mathbf{x}, N_j(\mathbf{x}))}$$

    Step 4. 이웃 인스턴스들의 평균 LRD와 쿼리 인스턴스의 LRD의 비율로 이상점수를 계산
    (쿼리 인스턴스의 국지적 밀도가 이웃의 국지적 밀도보다 상대적으로 낮으면 이상으로 판단)

    $$\text{LOF}_k(\mathbf{x}) = \frac{\sum_{j=1}^{k} \text{lrd}_k(N_j(\mathbf{x}))/k}{\text{lrd}_k(\mathbf{x})}$$

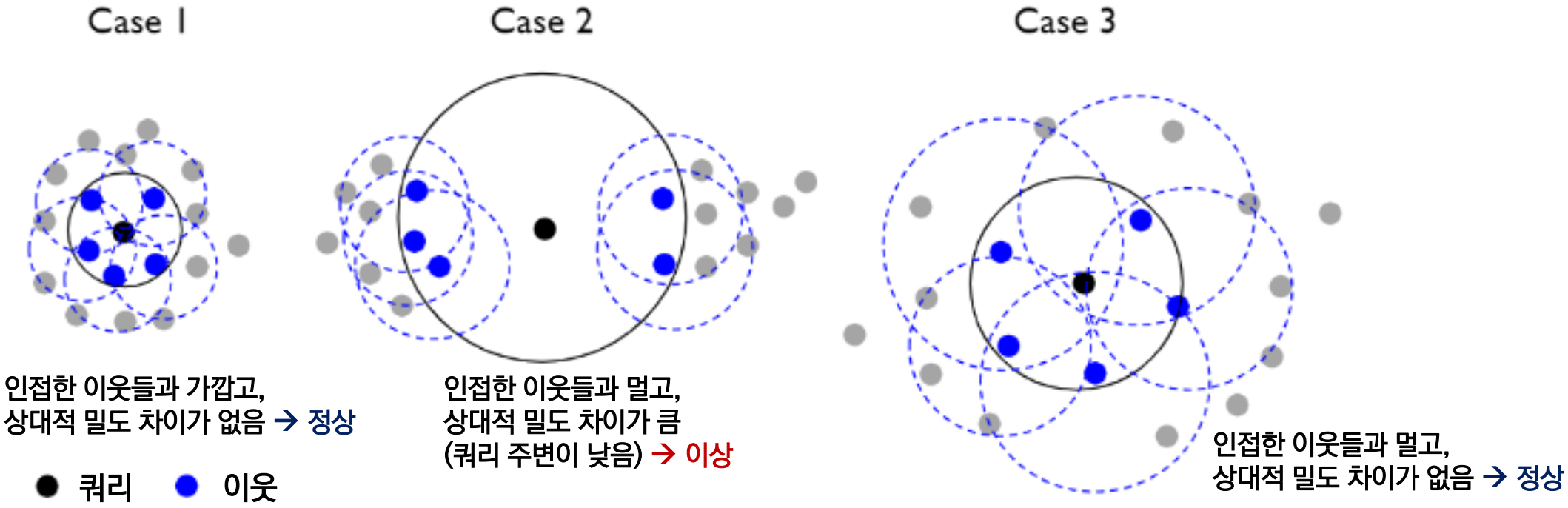
거리(Distance) 기반 이상탐지

- Local Outlier Factor (LOF): 쿼리 인스턴스 주변의 국지적 밀도(Local Density)가 학습 데이터셋 내 최근접 이웃 인스턴스들의 국지적 밀도보다 상대적으로 낮으면 이상으로 판단하는 방법
  - 예시: 인스턴스 B, C, D가 각각 쿼리 인스턴스일 때, A에 대한 Reachability Distance (k=3)
    - ✓ $d(\mathbf{D}, \mathbf{A}) > d(\mathbf{B}, \mathbf{A}) > d(\mathbf{C}, \mathbf{A})$
    - ✓ $\text{reach\_dist}_k(\mathbf{D}, \mathbf{A}) > \text{reach\_dist}_k(\mathbf{B}, \mathbf{A}) = \text{reach\_dist}_k(\mathbf{C}, \mathbf{A})$



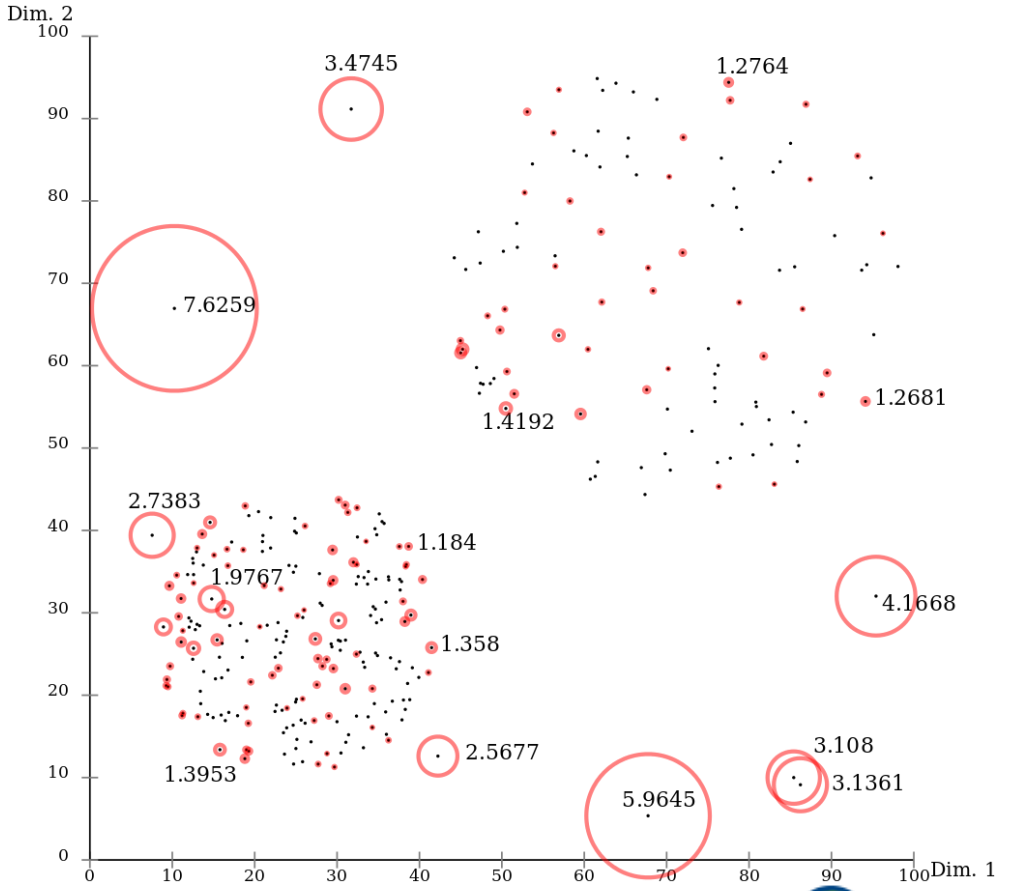
거리(Distance) 기반 이상탐지

- Local Outlier Factor (LOF): 쿼리 인스턴스 주변의 국지적 밀도(Local Density)가 학습 데이터셋 내 최근접 이웃 인스턴스들의 국지적 밀도보다 상대적으로 낮으면 이상으로 판단하는 방법
  - 예시: LOF의 이상 판단 케이스 (k=5)



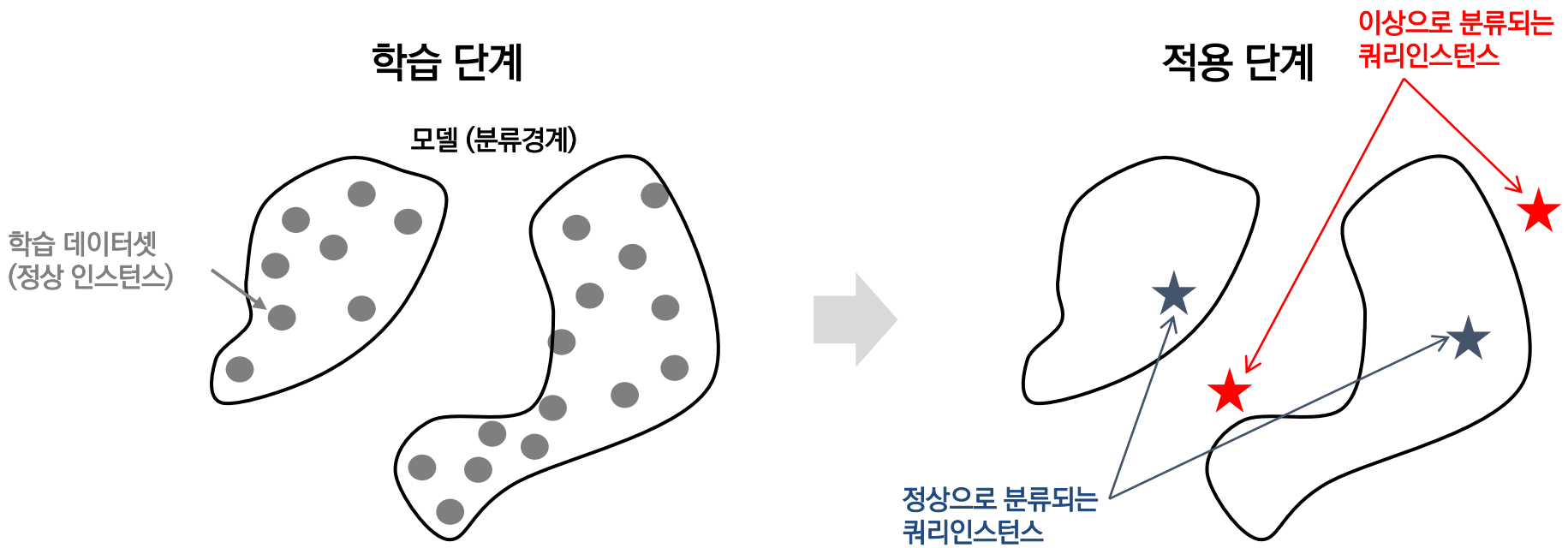
거리(Distance) 기반 이상탐지

- **Local Outlier Factor (LOF):** 쿼리 인스턴스 주변의 국지적 밀도(Local Density)가 학습 데이터셋 내 최근접 이웃 인스턴스들의 주변의 국지적 밀도보다 상대적으로 낮으면 이상으로 판단하는 방법
  - **이상점수에 따른 이상 여부 판단**
    - ✓ LOF < 1: 쿼리 인스턴스의 상대적 밀도가 이웃 인스턴스들 대비 높음 (정상)
    - ✓ LOF > 1: 쿼리 인스턴스의 상대적 밀도가 이웃 인스턴스들 대비 낮음 (이상)
  - **중요 하이퍼파라미터:** 쿼리 인스턴스의 이웃 개수 k (+ 인스턴스 간 거리의 기준)
    - ✓ 이상탐지 결과가 k의 설정에 크게 영향을 받음
    - ✓ 작을수록 노이즈/Outlier에 민감, 클수록 데이터의 지역적(Local) 특성을 무시



분류경계(Boundary) 기반 이상탐지

- **쿼리 인스턴스가 정상 데이터를 설명하는 분류경계 바깥에 위치하면 이상으로 판단**
  - 이상 데이터는 정상 데이터가 존재하는 영역의 바깥에 위치함을 가정
  - **학습 단계:** 학습 데이터셋을 포함하는 분류 경계를 직접적으로 형성하는 모델 학습
  - **적용 단계:** 주어진 쿼리 인스턴스 $\mathbf{x}$가 분류 경계로부터 바깥 방향으로 멀리 떨어져 있으면 높은 이상점수 부여



scikit-learn implementation:
sklearn.svm.OneClassSVM

분류경계(Boundary) 기반 이상탐지

- One-Class Support Vector Machine (OC-SVM): 특성 공간(Feature Space) 상에서 학습 데이터를 원점으로부터 분리하는 초평면(Hyperplane)으로 선형 분류경계를 형성하는 방법
  - **학습 단계에서 초평면 $\mathbf{w}^T\varphi(\mathbf{x}) - \rho = 0$의 원점으로부터의 거리(Margin)을 최대화하는 최적화 문제 설계 (Maximum-Margin Hyperplane)**
  - φ는 인스턴스 $\mathbf{x}$를 고차원의 특성 공간(Feature Space)으로 매핑($\mathbf{x} \rightarrow \varphi(\mathbf{x})$)하는 함수로, 적절한 함수를 도입하여 특성 공간에서의 선형 분류 경계가 원 공간에서 복잡한 비선형 분류 경계에 대응하도록 할 수 있음

    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$

margin을 최대화 / Trade-Off 하이퍼파라미터

$$\underset{\mathbf{w},\rho,\xi_i}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} - \rho + \frac{1}{\nu N}\sum_i \xi_i$$

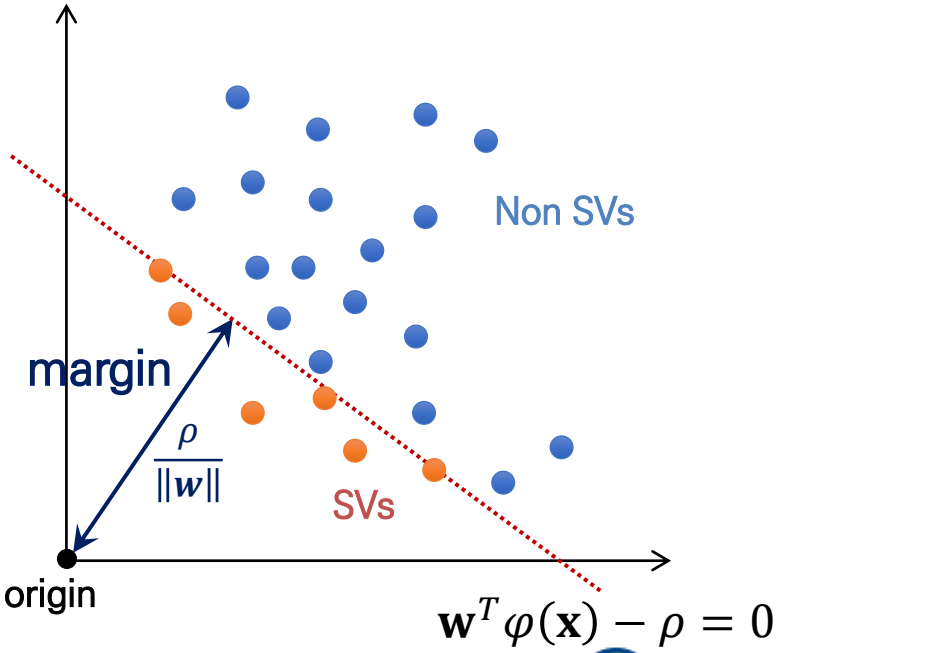
학습데이터에 대한 분류 오류를 최소화

$$\text{subject to} \quad \mathbf{w}^T\varphi(\boldsymbol{x}_i) \geqslant \rho - \xi_i,$$

일부 분류 오류를 허용

학습데이터가 원점으로부터 초평면에 의해 분리되도록 함

$$\xi_i \geqslant 0, i = 1, \ldots, N,$$



분류경계(Boundary) 기반 이상탐지

- One-Class Support Vector Machine (OC-SVM): 특성 공간(Feature Space) 상에서 학습 데이터를 원점으로부터 분리하는 초평면(Hyperplane)으로 선형 분류경계를 형성하는 방법
  - 원 최적화 문제(Primal Problem)를 직접 푸는 대신, 쌍대 문제(Dual Problem)에 대한 해를 도출
  - 쌍대문제는 볼록 최적화(Convex Optimization) 문제의 형태로, 전역 최적해(Global Optimum) 도출 가능
  - 커널 함수 $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$는 특성 공간 상에서 두 인스턴스 간의 내적 연산을 정의함
    - ✓ 학습 과정에서 실제 φ의 형태를 직접 알 필요가 없음
    - ✓ RBF Kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ 이 일반적으로 사용됨

$$\underset{\alpha_i}{\text{maximize}} \quad -\sum_{i,j} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{subject to} \quad \sum_i \alpha_i = 1,$$

$$0 \leqslant \alpha_i \leqslant \frac{1}{\nu N}, i = 1, \ldots, N,$$

분류경계(Boundary) 기반 이상탐지

- One-Class Support Vector Machine (OC-SVM): 특성 공간(Feature Space) 상에서 학습 데이터를 원점으로부터 분리하는 초평면(Hyperplane)으로 선형 분류경계를 형성하는 방법
  - **적용 단계에서 쿼리 인스턴스 $\mathbf{x}$ 에 대해서 아래와 같이 이상점수를 계산**

$$f(\mathbf{x}) = \boldsymbol{w}^T \varphi(\boldsymbol{x}) - \rho = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho$$

  - ✓ $f(\mathbf{x}) > 0$이면 인스턴스가 분류 경계에서 원점에 먼 방향에 위치함 → **정상**
  - ✓ $f(\mathbf{x}) < 0$이면 인스턴스가 분류 경계에서 원점에 가까운 방향에 위치함 → **이상**

  - Support Vector (SV)의 집합 $D_{SV} = \{\mathbf{x}_i \in D | \alpha_i > 0\}$을 활용하여 이상탐지 모델 f를 단순화 표현할 수 있음

$$f(\mathbf{x}) = \sum_{\mathbf{x}_i \in D_{SV}} \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho$$

  - ✓ Support Vector는 학습 데이터셋 중 $f(\mathbf{x}) \leq 0$을 만족하는 인스턴스로, 분류 경계 위 또는 원점에 가까운 방향에 위치함

분류경계(Boundary) 기반 이상탐지

- **Support Vector Data Description (SVDD):** 특성 공간(Feature Space) 상에서 학습 데이터를 포함하는 초구(Hypersphere)로 분류경계를 형성하는 방법
  - **학습 단계에서 초구 $\|\varphi(\mathbf{x}) - \mathbf{a}\|^2 = R^2$의 크기를 최소화하는 최적화 문제 설계 (Data-Enclosing Hypershpere)**
  - OC-SVM과 동일하게, φ는 인스턴스 $\mathbf{x}$를 고차원의 특성 공간으로 매핑($\mathbf{x} \rightarrow \varphi(\mathbf{x})$)하는 함수

    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$

초구의 크기를 최소화 Trade-Off 하이퍼파라미터

$$\underset{R,\boldsymbol{a},\xi_i}{\text{minimize}} \quad R^2 + \frac{1}{\nu N}\sum_i \xi_i$$

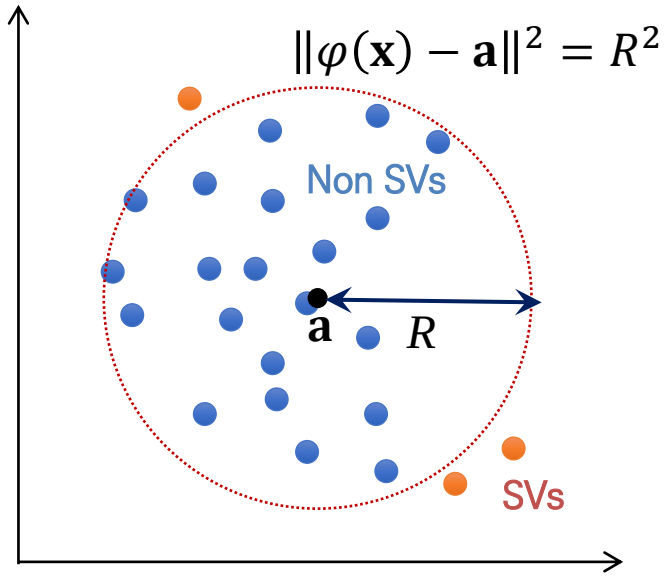
학습데이터에 대한 분류 오류를 최소화

$$\text{subject to} \quad \|\varphi(\boldsymbol{x}_i) - \mathbf{a}\|^2 \leqslant R^2 + \xi_i,$$

학습데이터가 초구 안에 위치하도록 함

$$\xi_i \geqslant 0, i = 1, \ldots, N,$$

일부 분류 오류를 허용



분류경계(Boundary) 기반 이상탐지

- Support Vector Data Description (SVDD): 특성 공간(Feature Space) 상에서 학습 데이터를 포함하는 초구(Hypersphere)로 분류경계를 형성하는 방법
  - 원 최적화 문제(Primal Problem)를 직접 푸는 대신, 쌍대 문제(Dual Problem)에 대한 해를 도출
  - 쌍대문제는 볼록 최적화(Convex Optimization) 문제의 형태로, 전역 최적해(Global Optimum) 도출 가능
  - 커널 함수 $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$는 특성 공간 상에서 두 인스턴스 간의 내적 연산을 정의함
    - ✓ 학습 과정에서 실제 φ의 형태를 직접 알 필요가 없음
    - ✓ RBF Kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ 이 일반적으로 사용됨

$$\underset{\alpha_i}{\text{maximize}} \quad \sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{subject to} \quad \sum_i \alpha_i = 1,$$

$$0 \leqslant \alpha_i \leqslant \frac{1}{\nu N}, i = 1, \ldots, N,$$

분류경계(Boundary) 기반 이상탐지

- **Support Vector Data Description (SVDD):** 특성 공간(Feature Space) 상에서 학습 데이터를 포함하는 초구(Hypersphere)로 분류경계를 형성하는 방법
  - **적용 단계에서 쿼리 인스턴스 $\mathbf{x}$ 에 대해서 아래와 같이 이상점수를 계산**

$$f(\mathbf{x}) = R^2 - \|\varphi(\boldsymbol{x}) - \mathbf{a}\|^2$$

    * $\|\varphi(\mathbf{x}) - \mathbf{a}\|^2 = k(\mathbf{x},\mathbf{x}) - 2\sum_i \alpha_i k(\mathbf{x}_i,\mathbf{x}) + \sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i,\mathbf{x}_j)$
    - ✓ $f(\mathbf{x}) > 0$이면 인스턴스가 분류 경계에서 구의 중심 $\mathbf{a}$에 가까운 방향에 위치함 → **정상**
    - ✓ $f(\mathbf{x}) < 0$이면 인스턴스가 분류 경계에서 구의 중심 $\mathbf{a}$에 먼 방향에 위치함 → **이상**

  - Support Vector (SV)의 집합 $D_{SV} = \{\mathbf{x}_i \in D | \alpha_i > 0\}$을 활용하여 이상탐지 모델 f를 단순화 표현할 수 있음
    - ✓ Support Vector는 학습 데이터셋 중 $f(\mathbf{x}) \leq 0$을 만족하는 인스턴스로, 분류 경계 위 또는 구의 중심에 가까운 방향에 위치함

분류경계(Boundary) 기반 이상탐지

- OC-SVM & SVDD
  - 커널 함수 $k(\mathbf{x}, \mathbf{x}')$가 $\|\mathbf{x} - \mathbf{x}'\|$의 함수이면, 즉 $\mathbf{x} = \mathbf{x}'$일 때 상수 값을 가지면, 두 방법론은 원 공간에서 완전히 동일한 분류 경계를 형성함
  - 예시: RBF Kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$

OC-SVM의 쌍대 최적화문제

$$\underset{\alpha_i}{\text{maximize}} \quad -\sum_{i,j} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{subject to} \quad \sum_i \alpha_i = 1,$$

$$0 \leqslant \alpha_i \leqslant \frac{1}{\nu N}, i = 1, \ldots, N,$$

SVDD의 쌍대 최적화문제

$$\underset{\alpha_i}{\text{maximize}} \quad \sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{subject to} \quad \sum_i \alpha_i = 1,$$

$$0 \leqslant \alpha_i \leqslant \frac{1}{\nu N}, i = 1, \ldots, N,$$

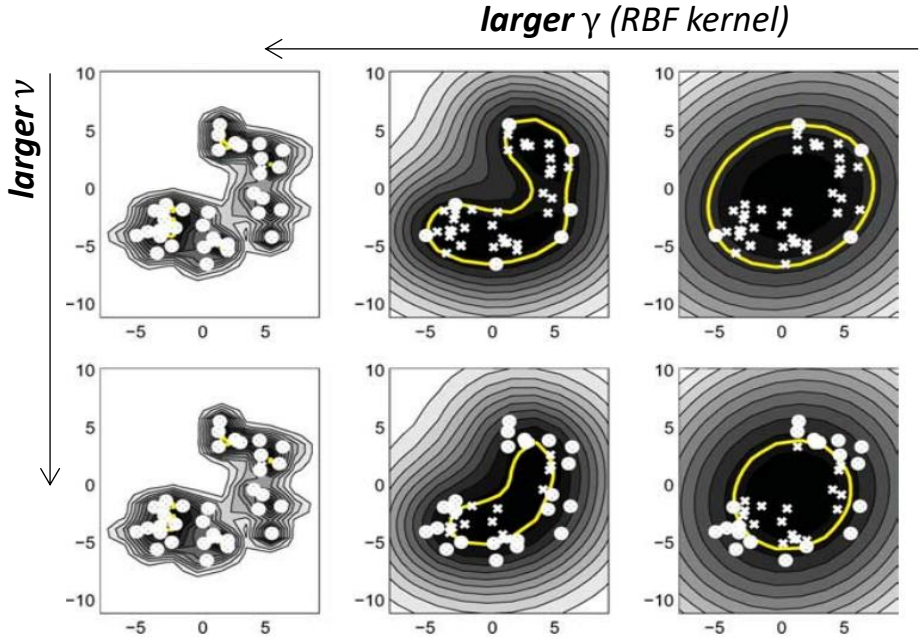
이상탐지 함수 간 관계

$$\begin{aligned}
f_{SVDD}(\boldsymbol{x}) &= R^2 - d^2(\boldsymbol{x}) \\
&= \left(k(\boldsymbol{x}_{\mathrm{SV}}, \boldsymbol{x}_{\mathrm{SV}}) - 2\sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_{\mathrm{SV}}) + \sum_{i,j} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \\
&\quad - \left(k(\boldsymbol{x}, \boldsymbol{x}) - 2\sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \sum_{i,j} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \\
&= 2\sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - 2\sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_{\mathrm{SV}}) \\
&= 2\left(\sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho \right) \\
&= 2 f_{OCSVM}(\boldsymbol{x})
\end{aligned}$$

분류경계(Boundary) 기반 이상탐지

- OC-SVM & SVDD
  - **중요 하이퍼파라미터:** Trade-Off 하이퍼파라미터 ν, 커널 함수의 하이퍼파라미터 γ (RBF Kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$ 사용 시)
    - ✓ ν가 클수록 분류 경계의 학습 데이터에 대한 오류를 더 허용하여 분류경계가 설명하는 영역이 작아짐
    - ✓ γ가 클수록 개별 인스턴스의 분류 경계에 대한 영향력이 작아지며 따라서 분류 경계가 복잡해짐

하이퍼파라미터 설정에 따른 원 공간에서의 분류 경계

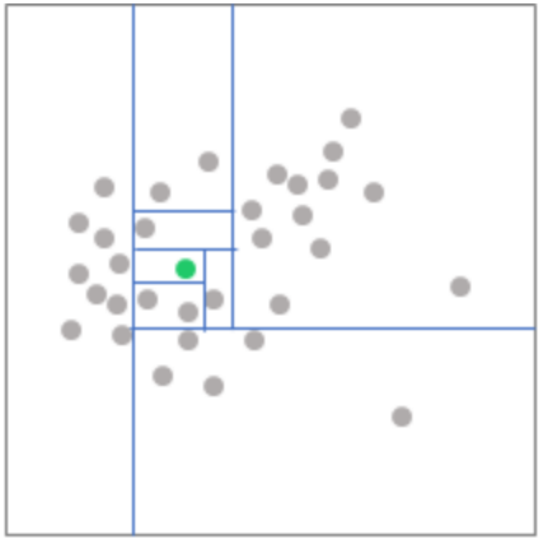


scikit-learn implementation:
sklearn.ensemble.IsolationForest

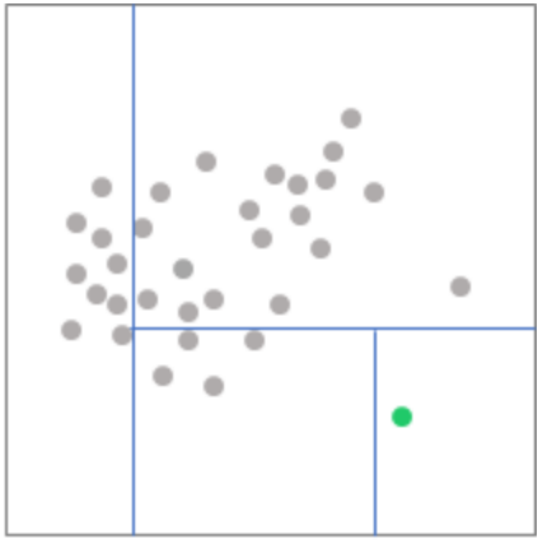
분류경계(Boundary) 기반 이상탐지

- Isolation Forest: 학습 데이터로부터 쉽게 고립되는 위치에 있는 쿼리 인스턴스를 이상으로 판단
  - 이상 인스턴스는 정상 인스턴스 대비 나머지 데이터로부터 고립시키기가 쉬움을 가정
  - **학습 단계에서 축에 평행한 선형 분류경계들을 활용하여 학습 데이터의 개별 인스턴스를 고립시키도록 하는 의사결정나무인 iTree를 여러 개 학습**
    * iTree의 개수는 일반적으로 100개로 설정

정상 인스턴스의 고립 (7번의 split)



이상 인스턴스의 고립 (3번의 split)



분류경계(Boundary) 기반 이상탐지

- Isolation Forest: 학습 데이터로부터 쉽게 고립되는 위치에 있는 쿼리 인스턴스를 이상으로 판단
  - **개별 iTree의 학습 절차 (학습 데이터셋의 일부를 "무작위"로 고립하도록 학습)**

    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$

    Step 1. 전체 학습 데이터셋에 대한 부분집합 $S \in D$를 임의로 추출

    * 일반적으로 부분집합 S의 크기는 256으로 설정

    Step 2. 부분집합 S로 구성되는 루트 노드 생성

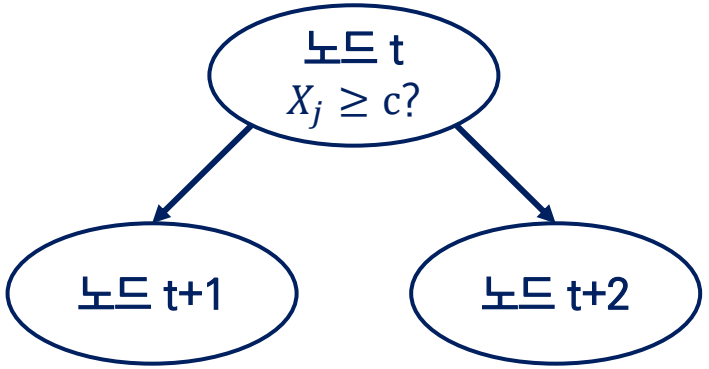
    Step 3. 모든 노드가 분할 종료 조건에 도달할 때까지 각 노드에 대해 아래의 과정을 반복

    * 분할 종료 조건: 노드에 하나의 인스턴스만 고립되거나,
      노드의 경로 길이(루트 노드로부터의 거리)가 최대 허용치에 도달

    2-1. 하나의 변수 X_j를 임의로 선택

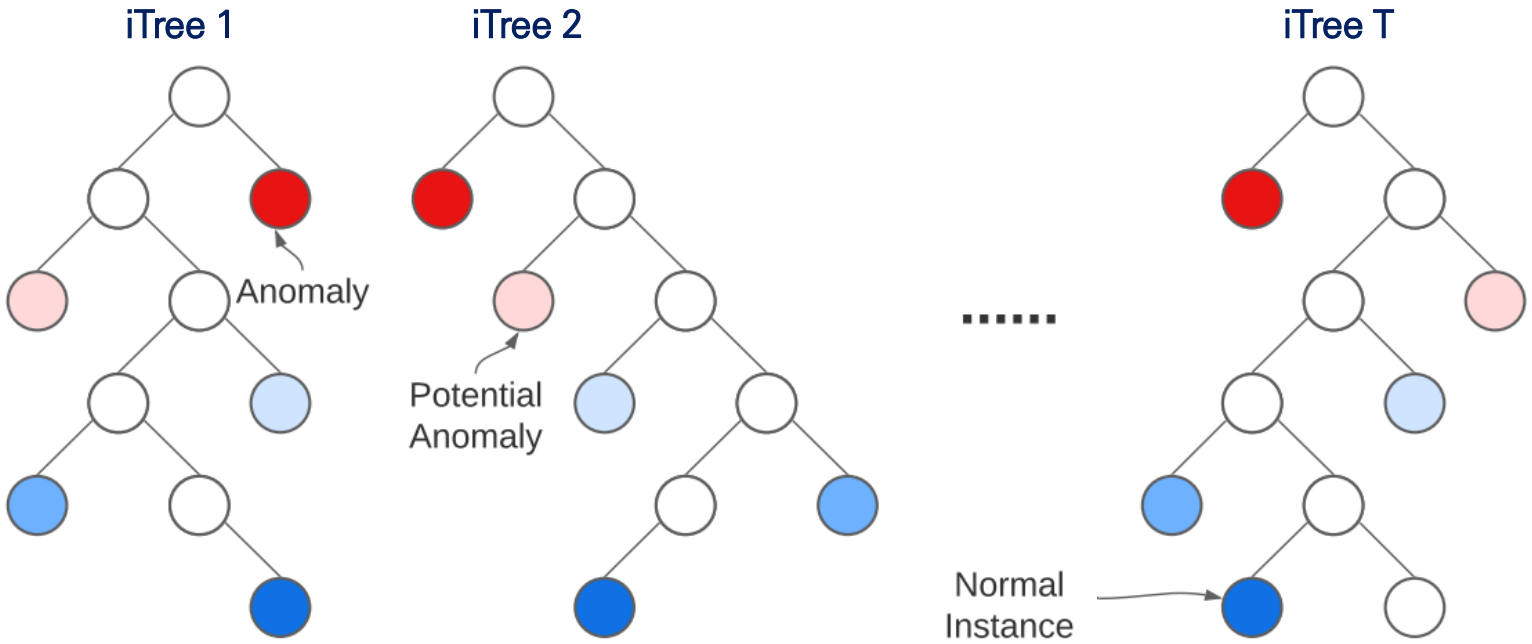
    2-2. 노드에서 관측된 X_j의 최대값과 최소값 사이에서 임의의 분할점 c를 선택

    2-3. 선택된 변수의 분할점을 기준으로 노드를 2개의 노드로 분리 ($X_j \geq c$)



분류경계(Boundary) 기반 이상탐지

- Isolation Forest: 학습 데이터로부터 쉽게 고립되는 위치에 있는 쿼리 인스턴스를 이상으로 판단
  - **적용 단계에서 쿼리 인스턴스 x에 대한 이상점수 계산 절차**
    - ✓ 쿼리 인스턴스를 개별 iTree를 이용하여 분류 시, 루트 노드에서부터 최종 도달하는 터미널 노드까지의 경로 길이(path length)를 구함
    - ✓ 전체 iTree에 대한 평균 경로 길이가 짧을수록 높은 이상점수 부여



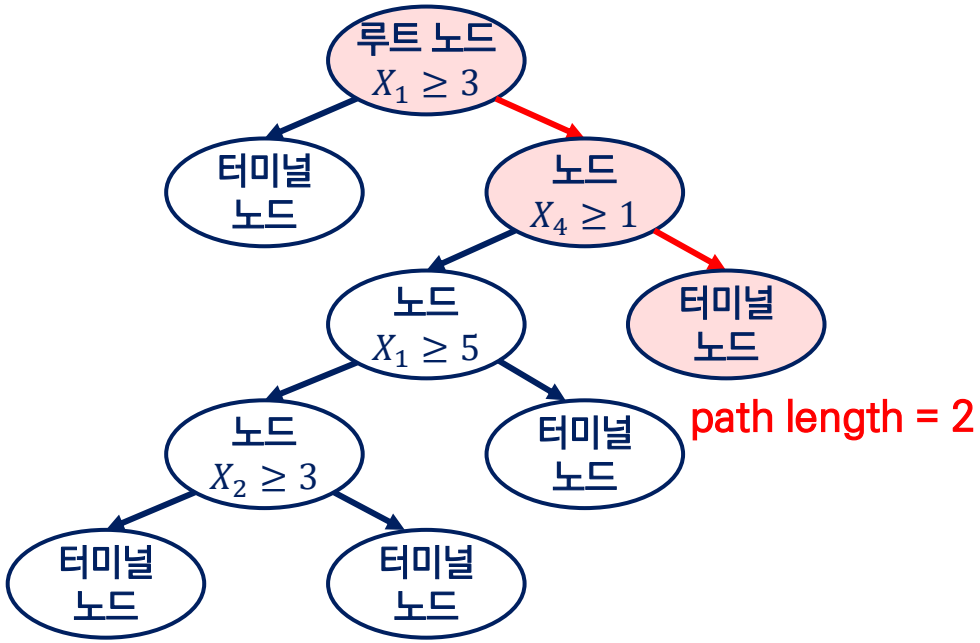
Case 1
- iTree 1의 path length = 1
- iTree 2의 path length = 1
  ...
- iTree T의 path length = 1

Case 2
- iTree 1의 path length = 5
- iTree 2의 path length = 5
  ...
- iTree T의 path length = 5

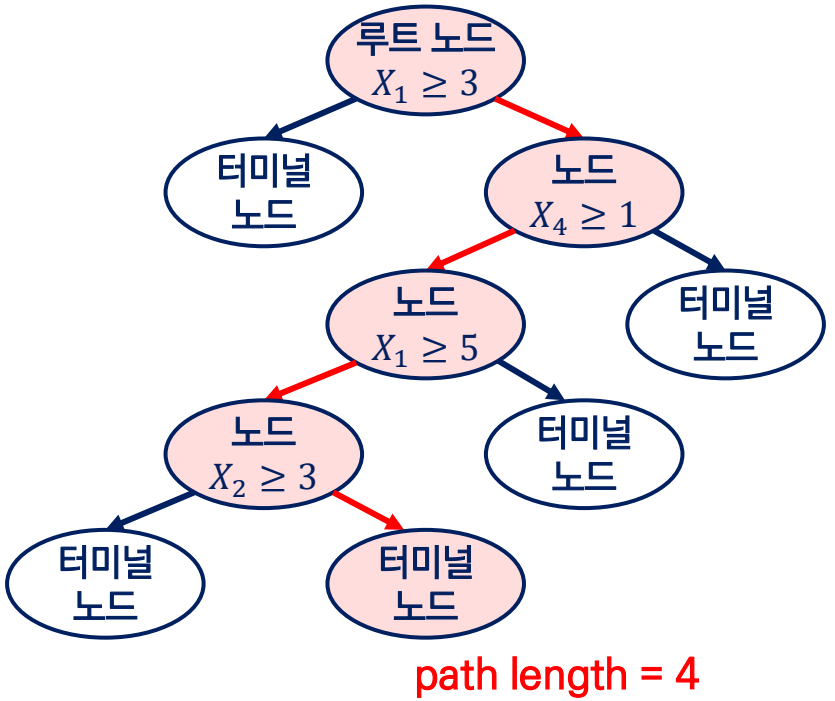
분류경계(Boundary) 기반 이상탐지

- Isolation Forest: 학습 데이터로부터 쉽게 고립되는 위치에 있는 쿼리 인스턴스를 이상으로 판단
  - 예시: iTree를 사용한 쿼리 인스턴스의 분류

쿼리 인스턴스 $\mathbf{x} = (5, 4, 3, 2)$

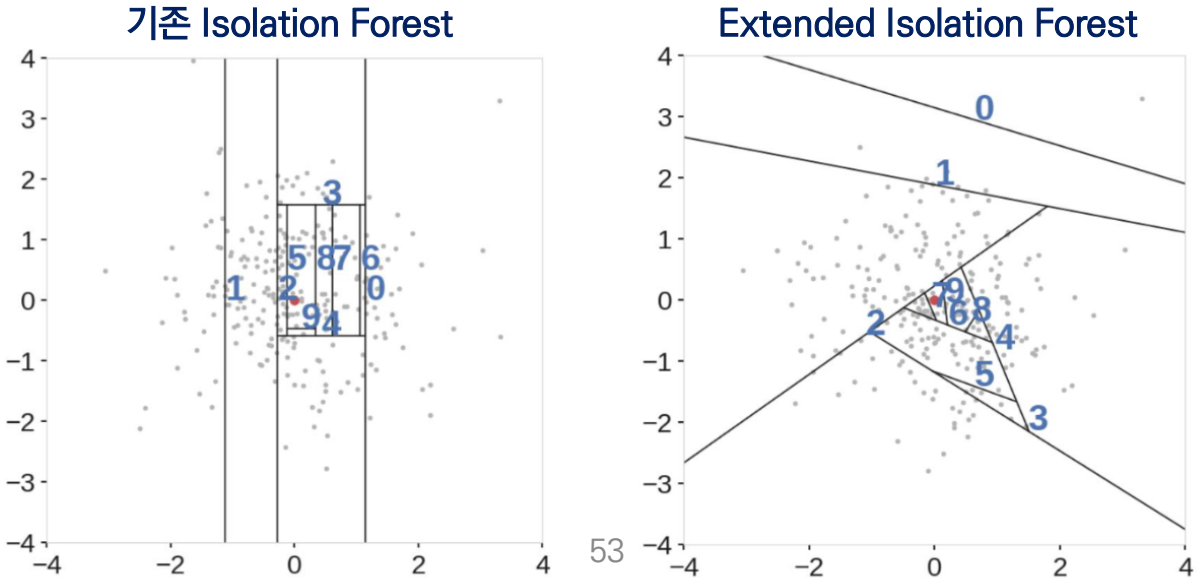


쿼리 인스턴스 $\mathbf{x} = (4, 4, 3, 0)$



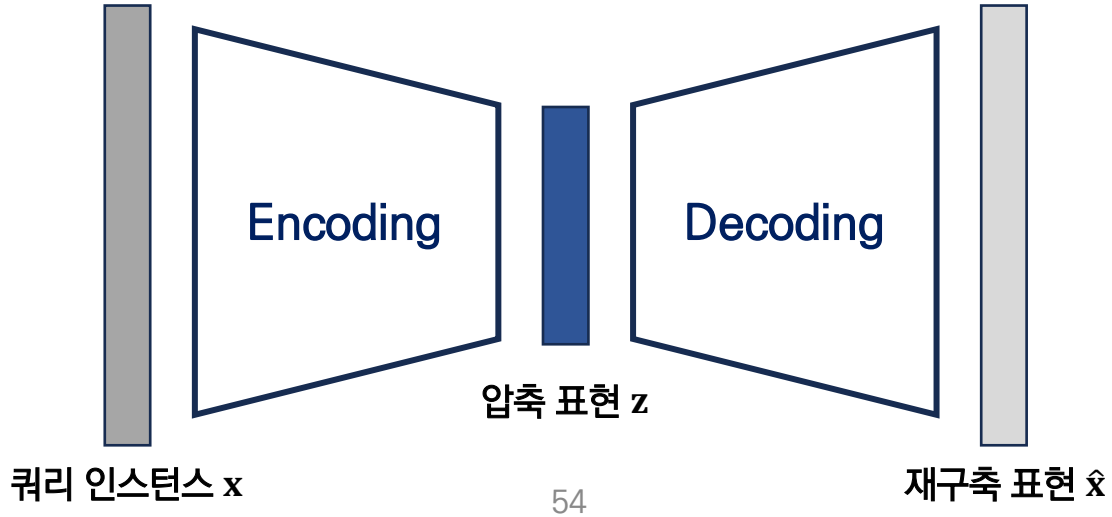
분류경계(Boundary) 기반 이상탐지

- Isolation Forest: 학습 데이터로부터 쉽게 고립되는 위치에 있는 쿼리 인스턴스를 이상으로 판단
  - 모델의 학습 및 추론 속도가 빠름
  - 인스턴스 간 거리를 활용하지 않으므로 변수 Scaling과 거리지표에 대한 고려 불필요
  - 하이퍼파라미터 설정에 강건하여 기본 설정을 그대로 사용해도 좋은 성능을 얻을 수 있음
    - ✓ iTree 개수 = 100, 학습 부분집합 크기 = 256
  - Isolation Forest의 개선: Extended Isolation Forest
    - ✓ 축에 평행하지 않은 선형 분류경계를 활용하는 iTree (Rotated iTree)를 활용



재구축(Reconstruction) 기반 이상탐지

- **쿼리 인스턴스를 재구축 모델을 이용하여 압축한 후 다시 복원했을 때 원래의 표현과 많이 달라지는 경우 이상으로 판단**
  - 데이터 표현의 압축 시 정상과 이상 데이터 간 추출되는 주요 특성이 다름을 가정
  - **학습 단계:** 학습 데이터셋을 인스턴스를 압축했다가 복원하는 재구축 모델 학습
  - **적용 단계:** 주어진 쿼리 인스턴스 $\mathbf{x}$가 재구축 모델을 이용하여 복원이 잘 되지 않을수록 높은 이상점수 부여
  - 주의: 재구축 모델의 압축과 재구축 간 Trade-Off
    - ✓ 이상탐지 관점에서 재구축 모델이 주어진 인스턴스를 정확하게 재구축을 하는 것이 주 목표가 아님
    - ✓ 압축 표현이 너무 작으면/크면 정상과 이상 인스턴스 모두에 대해서 재구축을 못하게/잘하게 됨



scikit-learn implementation:
sklearn.decomposition.PCA

재구축(Reconstruction) 기반 이상탐지

- **Principal Component Analysis (PCA): 재구축 모델로 PCA를 활용**
  - PCA는 기존 데이터의 분산을 최대한 보존하는 새로운 축을 찾고, 그 축으로 데이터를 사영(Projection)하는 방법으로, 주로 차원축소의 용도로 사용되나 재구축 모델의 용도로도 사용될 수 있음
    - ✓ 새로운 축은 기존 변수$(X_1, X_2, \dots, X_d)$의 선형 조합으로 표현되는 새로운 변수$(Z_1, Z_2, \dots, Z_d)$에 대응됨
    - ✓ 새로운 변수$(Z_1, Z_2, \dots, Z_d)$는 서로 상관관계가 없으며, 새로운 변수 중 일부가 원 데이터의 대부분 정보를 표현
    - ✓ 주의: 기존 변수에 대한 적절한 스케일링 필요 – 스케일이 더 큰 변수가 더 많은 정보를 가지고 있는 것으로 간주됨.

  - **학습단계에서 원래 데이터 차원 d보다 적은 수의 r개의 주성분 벡터 추출**

    * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 평균 벡터 $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$

    Step 1. 학습 데이터셋의 공분산 행렬 $S = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$ 계산

    Step 2. 공분산 행렬 S의 고유값 분해(Eigendecomposition)을 통해 d개의 고유벡터(Eigenvector) $\mathbf{u}_1, \dots, \mathbf{u}_d$와 그에 대응되는 고유값(Eigenvalue) $\lambda_1, \dots, \lambda_d$ 도출
    (고유벡터는 고유값의 크기 순으로 내림차순 정렬함을 가정: $\lambda_1 > \lambda_2 > \cdots > \lambda_d$)

    Step 3. 고유값이 큰 순서로 r개의 고유벡터 $\mathbf{u}_1, \dots, \mathbf{u}_r$를 저장 → **주성분 (Principal Component) 벡터**

재구축(Reconstruction) 기반 이상탐지

- Principal Component Analysis (PCA): 재구축 모델로 PCA를 활용
  - PCA의 원리에 대한 수리적인 설명
    - ✓ 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$의 각 인스턴스에 대한 사영(Projection)은 다음과 같이 표현될 수 있음

$$z_i = \mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}})$$

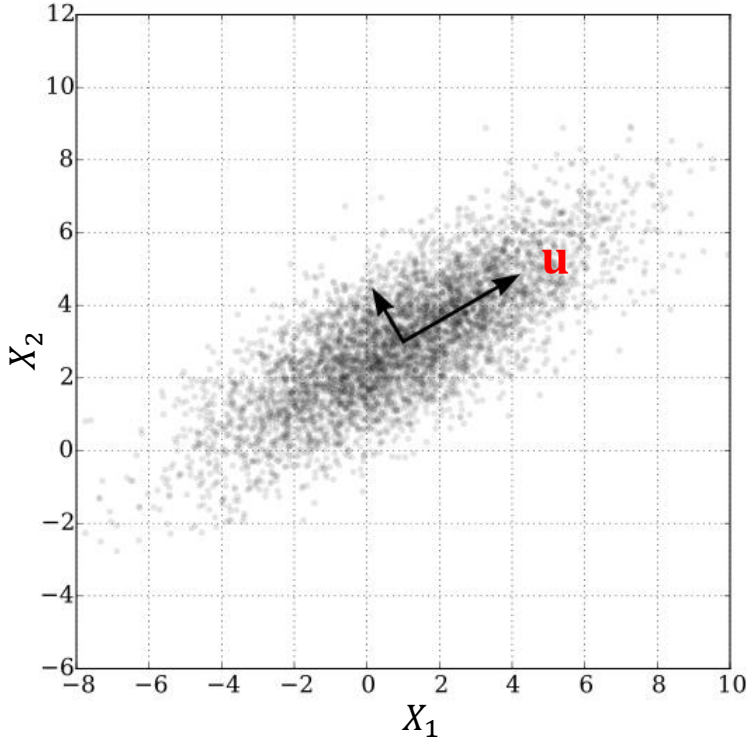
    - ✓ 학습 데이터셋의 사영 $\{z_1, z_2, \dots, z_n\}$의 분산을 최대화하는 벡터 $\mathbf{u}$를 찾고자 함

$$\frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{u}^T\mathbf{x}_i - \mathbf{u}^T\bar{\mathbf{x}}\right)^2 = \mathbf{u}^T\mathbf{S}\mathbf{u}$$

    - ✓ 아래와 같은 최적화 문제를 설계

$$\max_{\mathbf{u}} \ \mathbf{u}^T\mathbf{S}\mathbf{u}$$
$$\text{s.t. } \mathbf{u}^T\mathbf{u} = 1$$

      - 제약식은 벡터 $\mathbf{u}$의 크기(L2 norm)를 1 (단위벡터)로 함
      - 벡터 $\mathbf{u}$의 크기에 대한 효과를 제거하고 최적의 방향을 탐색
    - ✓ 위 최적화 문제에 대한 해법은 공분산 행렬 $\mathbf{S}$에 대한 고유값 분해와 동일함
      - 총 d개의 고유벡터 $\mathbf{u}_1, \dots, \mathbf{u}_d$와 그에 대한 고유값 $\lambda_1, \dots, \lambda_d$이 존재하며, 모든 고유벡터는 서로 독립
      - 각 고유벡터의 고유값은 해당 벡터를 이용한 학습 데이터셋의 사영이 설명하는 분산
      - 전체 고유값의 합 $\sum_{j=1}^{d}\lambda_j$은 학습 데이터셋에 대한 각 변수의 분산 합 (공분산 행렬 S의 Trace – 대각 원소의 합)과 같음



재구축(Reconstruction) 기반 이상탐지

- Principal Component Analysis (PCA): 재구축 모델로 PCA를 활용
  - **적용단계에서 쿼리 인스턴스 $\mathbf{x}$에 대한 이상점수 계산 절차**

    * r개의 주성분 벡터 $\mathbf{u}_1, \ldots, \mathbf{u}_r$ 활용

    Step 1. 쿼리 인스턴스 $\mathbf{x}$에 대한 압축 (Encoding)

    $$\mathbf{z} = (z_1, \ldots, z_r), \qquad z_j = \mathbf{u}_j^T(\mathbf{x} - \bar{\mathbf{x}})$$

    Step 2. 압축 표현 $\mathbf{z}$의 복원 (Decoding)

    $$\hat{\mathbf{x}} = \sum_{j=1}^{r} z_j \mathbf{u}_j + \bar{\mathbf{x}} = \sum_{j=1}^{r} (\mathbf{u}_j^T \mathbf{x})\mathbf{u}_j + \bar{\mathbf{x}}$$

    Step 3. 재구축 오차를 이상점수로 활용

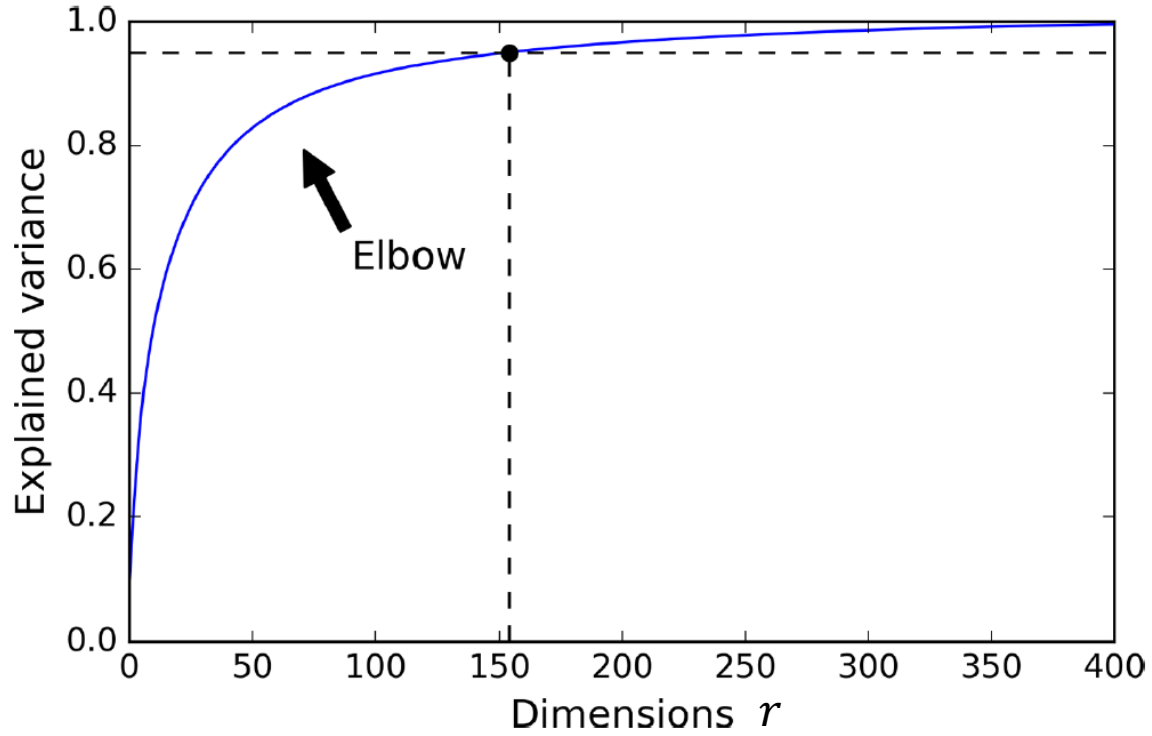
    $$f(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|$$

    ✓ $f(\mathbf{x})$가 클수록 인스턴스에 대한 재구축 오차가 큼 → 이상

재구축(Reconstruction) 기반 이상탐지

- Principal Component Analysis (PCA): 재구축 모델로 PCA를 활용
  - **중요 하이퍼파라미터:** 압축 표현의 차원 r (사용하는 주성분 벡터의 개수)
    - ✓ 초기 몇 개의 주성분 벡터가 전체 분산의 많은 비중을 설명하며, r이 커짐에 따라 분산 설명 비율의 증가 기울기가 감소
    - ✓ 일반적으로, 일정 수준 이상(예: 90%)의 분산 설명 비율을 만족하도록 하는 r 값을 선택

$$\frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$$



재구축(Reconstruction) 기반 이상탐지

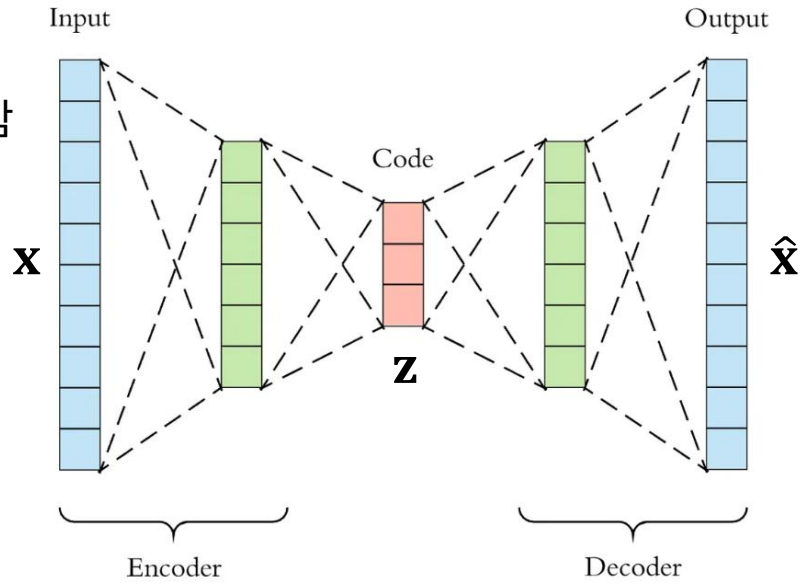
- **Autoencoder:** 재구축 모델로 인공신경망을 활용
  - 일반적인 인공신경망은 레이블을 예측하도록 학습이 되나, Autoencoder 인공신경망은 입력 인스턴스를 그대로 복원하도록 학습이 됨
    - ✓ 인공신경망 중간의 Bottleneck 레이어에서 입력 인스턴스에 대한 압축된 표현을 얻음
  - **학습 단계에서 학습 데이터셋에 대한 평균 재구축 오차를 최소화하도록 인공신경망을 학습**
    - * 학습 데이터셋 $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, 손실함수(Loss function) L

$$\frac{1}{n}\sum_{i=1}^{n} L(\mathbf{x}_i, \hat{\mathbf{x}}_i)$$

    - ✓ Bottleneck 레이어에서 압축 표현이 원래 인스턴스 정보를 최대한 보존하도록 함
  - **적용 단계에서 쿼리 인스턴스 $\mathbf{x}$에 대한 재구축 오차를 이상점수로 활용**

$$f(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|$$

    - ✓ $f(\mathbf{x})$가 클수록 인스턴스에 대한 재구축 오차가 큼 → 이상
  - Tabular Data 뿐 아니라 이미지, 텍스트 등 다양한 데이터 표현에 활용될 수 있음
    - ✓ CNN Autoencoder, RNN Autoencoder, ...
  - 중요 하이퍼파라미터: 인공신경망 크기, Bottleneck 레이어의 차원

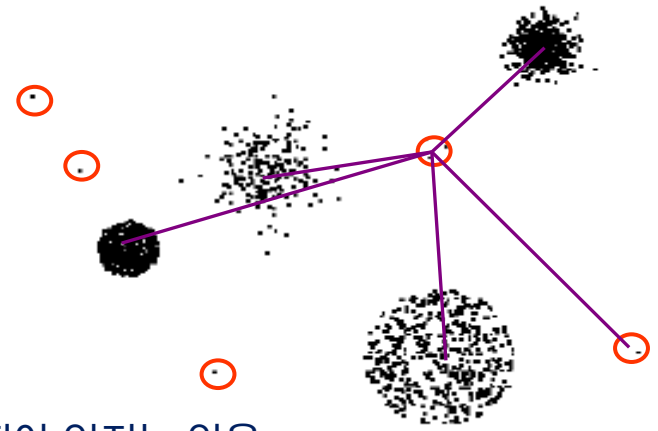


군집화(Clustering) 기반 이상탐지

- **쿼리 인스턴스가 학습 데이터로부터 형성된 어떠한 군집과도 가깝지 않으면 이상으로 판단**
  - 정상 데이터가 이상 데이터와 구분되는 여러 개의 군집을 형성함을 가정
  - **군집화(Clustering):** 주어진 학습 데이터의 인스턴스들을 유사한 특성을 갖는 몇 개의 군집으로 분할하는 과업
    - ✓ 군집은 동질적인 인스턴스들의 모임
    - ✓ 군집 간 거리는 최대화, 군집 내 인스턴스 간 거리는 최소화
  - **학습 단계:** 학습 데이터셋에 대한 군집화를 수행하여 군집 도출
    - ✓ 모든 군집화 알고리즘을 사용할 수 있음 (K-Means, Hierarchical, DBSCAN, …)
  - **적용 단계:** 주어진 쿼리 인스턴스 $\mathbf{x}$가 모든 군집과 멀리 떨어져 있을수록, 또는 매우 작은 군집(학습 데이터셋 내 이상 인스턴스로 추정)과 가까이 있는 경우, 높은 이상점수 부여
  - 예시: K-Means 군집화 기반 이상탐지
    - ✓ 학습 단계에서 K개의 군집에 대한 centroid vector $\mathbf{c}_1, \ldots, \mathbf{c}_K$ 도출
    - ✓ 적용 단계에서 가장 가까운 centroid vector와의 거리를 이상점수로 활용

$$f(\mathbf{x}) = \min_j \|\mathbf{x} - \mathbf{c}_j\|$$

  - 주의: 이상탐지 성능이 군집화 결과에 크게 의존함
    - ✓ 적절한 군집화 알고리즘 선택과 함께 적절한 하이퍼파라미터 설정이 중요
    - ✓ 군집화 알고리즘을 이상탐지에 활용할 수 있으나, 일반적으로 이상탐지 문제에 최적화 되어 있지는 않음

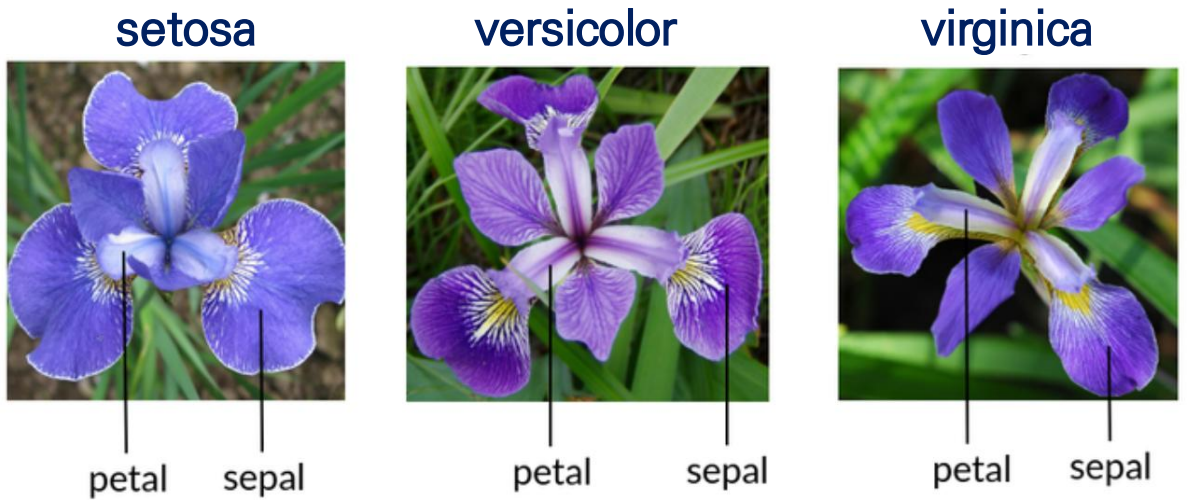


고려사항

- **이상탐지 문제에서 학습 데이터에 이상 인스턴스가 존재하지 않는 상황에 사용될 수 있음**
- **이상 인스턴스의 수가 극단적으로 적은 상황 (예를 들어, 10개 이내) 에도 지도학습 분류 접근 대비 효과적일 수도 있음**
- **이상 인스턴스가 존재하지 않기에, 하이퍼파라미터 최적화가 매우 어려움**
  - 현실 상황에서 Isolation Forest와 같이 데이터에 대한 가정이 크지 않으면서 하이퍼파라미터에 대한 민감성이 낮은 알고리즘이 선호됨
- **도메인 지식을 반영한 적절한 변수 선택 및 스케일링, 거리 지표 선택이 매우 중요**
  - 지도학습 방법론들은 일반적으로 학습 데이터에서 레이블의 예측을 위해 중요한 변수를 데이터 기반으로 선별하여 활용하나, 비지도학습 방법론들은 학습 데이터 표현 그 자체를 활용함

이상탐지 방법론 구현 예제

- **예제 문제** – Iris 데이터셋
  - Iris는 3개의 범주(setosa, versicolor, virginica)에 대해 각 50개의 인스턴스로, 총 150개의 인스턴스로 구성된 분류 벤치마크 데이터셋
  - 각 인스턴스는 4개의 변수 (1) the length of the petals, (2) the width of the petals, (3) the length of the sepals, and (4) the width of the sepals 에 대한 측정값으로 표현됨
  - 이상탐지 문제로, virginica 품종을 정상으로, 나머지 품종을 이상으로 간주하는 이상탐지 문제 설계
  - 전체 데이터셋을 학습과 평가의 목적으로 80:20로 나누고, 학습 데이터셋(Training Dataset)에는 virginica 품종 인스턴스만, 평가 데이터셋(Test dataset에는 모든 품종의 인스턴스가 존재하도록 실험 설계



이상탐지 방법론 구현 예제

- **sklearn을 활용한 예제 코드 –** Gaussian Density Estimation을 활용한 이상탐지

In [1]

```
# 필요한 패키지 모듈 임포트
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score

from sklearn.covariance import EllipticEnvelope
```

In [2]

```
# 예제 데이터셋 로드 및 이상탐지 문제 설정 (Class 2를 정상(0), Class 0&1을 이상(1))
iris = load_iris()
iris.target = (iris.target != 2) + 0
X_trn, X_tst, y_trn, y_tst = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=123, shuffle=True,
stratify=iris.target
)
X_trn = X_trn[y_trn==0]
```

이상탐지 방법론 구현 예제

- **sklearn을 활용한 예제 코드 –** Gaussian Density Estimation을 활용한 이상탐지

In [3]
```
# 변수 스케일링 (학습 데이터셋 기준 평균이 0, 분산이 1)
scaler = StandardScaler()
X_trn = scaler.fit_transform(X_trn)
X_tst = scaler.transform(X_tst)
```

In [4]
```
# 학습 데이터셋을 이용한 모델 학습 및 평가 데이터셋에 대한 이상점수 도출
model = EllipticEnvelope()
model.fit(X_trn)
y_tst_score = - model.score_samples(X_tst)
```

In [5]
```
#평가 결과 AUROC 출력
print('AUROC: %.4f'%roc_auc_score(y_tst, y_tst_score))
```

이상탐지 방법론 구현 예제

- **sklearn을 활용한 예제 코드** – 다른 이상탐지 방법론을 활용하고자 한다면?
  - 앞의 예제 코드에서 In [4]를 수정

Mixture of Gaussians (MOG)

```
from sklearn.mixture import GaussianMixture
model = GaussianMixture(n_components=5)
model.fit(X_trn)
y_tst_score = – model.score_samples(X_tst)
```

Parzen Window

```
from sklearn.neighbors import KernelDensity
model = KernelDensity(bandwidth = 1)
model.fit(X_trn)
y_tst_score = – model.score_samples(X_tst)
```

Isolation Forest (ISOF)

```
from sklearn.ensemble import IsolationForest
model = IsolationForest()
model.fit(X_trn)
y_tst_score = – model.score_samples(X_tst)
```

이상탐지 방법론 구현 예제

- **sklearn을 활용한 예제 코드** – 다른 이상탐지 방법론을 활용하고자 한다면?
  - 앞의 예제 코드에서 In [4]를 수정

One-Class Support Vector Machine (OCSVM)

```
from sklearn.svm import OneClassSVM
model = OneClassSVM(nu = 0.5, gamma = 1/4)
model.fit(X_trn)
y_tst_score = - model.score_samples(X_tst)
```

K-Nearest Neighbor (KNN)

```
from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors(n_neighbors = 5)
model.fit(X_trn)
distances, _ = model.kneighbors(X_tst)
y_tst_score = distances[:,-1]
```

Local Outlier Factor (LOF)

```
from sklearn.neighbors import LocalOutlierFactor
model = LocalOutlierFactor(n_neighbors=20, novelty=True)
model.fit(X_trn)
y_tst_score = - model.score_samples(X_tst)
```

이상탐지 방법론 구현 예제

- **sklearn을 활용한 예제 코드** – 다른 이상탐지 방법론을 활용하고자 한다면?
  - 앞의 예제 코드에서 In [4]를 수정

Principal Component Analysis (PCA)

```
from sklearn.decomposition import PCA
model = PCA(n_components = 0.9)
model.fit(X_trn)
y_tst_score = np.linalg.norm(X_tst - model.inverse_transform(model.transform((X_tst)), axis=1)
# y_tst_score = - model.score_samples(X_tst)
```

Autoencoder (AE)

```
from sklearn.neural_network import MLPRegressor
model = MLPRegressor(hidden_layer_sizes = 50, activation='tanh', solver='lbfgs')
model.fit(X_trn, X_trn)
y_tst_score = np.linalg.norm(X_tst - model.predict(X_tst), axis=1)
```

K-Means Clustering

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=5)
model.fit(X_trn)
y_tst_score = np.min(model.transform(X_tst), 1)
```

이상탐지 방법론 구현 예제

- **총 20개의 이상탐지 벤치마크 문제에 대한 성능 비교 결과** (평가지표: AUROC)
  - 비교된 10개의 이상탐지 방법론 중 Isolation Forest가 평균적으로 가장 높은 성능을 보여주었음
  - 벤치마크 문제마다 최적의 방법론은 다르며, 모든 문제에 대해 일관되게 우수한 성능을 가지는 방법론은 없음
  - 주의: 이상탐지 성능은 사용한 방법론 뿐 아니라 하이퍼파라미터 설정에 크게 영향을 받으며, 각 벤치마크 문제 별 개별 방법론의 성능은 하이퍼파라미터 설정을 최적화하여 더 개선될 수 있음

Method	scikit-learn class	Main hyperparameters	Setting
Gaussian	`sklearn.covariance.EllipticEnvelope`	-	-
Mixture of Gaussians (MOG)	`sklearn.mixture.GaussianMixture`	n_components	5
Parzen Window	`sklearn.neighbors.KernelDensity`	bandwidth	1 (default)
K-Nearest Neighbor (KNN)	`sklearn.neighbors.NearestNeighbors`	n_neighbors, metric	5, Euclidean (default)
Local Outlier Factor (LOF)	`sklearn.neighbors.LocalOutlierFactor`	n_neighbors, metric	20, Euclidean (default)
Isolation Forest (ISOF)	`sklearn.ensemble.IsolationForest`	n_estimators, max_samples	100, 256 (default)
K-Means Clustering (KMEANS)	`sklearn.cluster.KMeans`	n_clusters	5
Principal Component Analysis (PCA)	`sklearn.decomposition.PCA`	n_components	0.9
Autoencoder (AE)	`sklearn.neural_network.MLPRegressor`	hidden_layer_sizes, activation	50, tanh
One-Class Support Vector Machine (OCSVM)	`sklearn.svm.OneClassSVM`	nu, gamma	0.5, $1/d$ (default)

Problem	Gaussian	MOG	Parzen	KNN	LOF	ISOF	KMEANS	PCA	AE	OCSVM
iris-setosa	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9997	**1.0000**
iris-versicolor	**0.9916**	0.9727	0.9561	0.9804	0.9564	0.9760	0.9786	0.8987	0.9353	0.9710
iris-virginica	0.9387	0.9255	0.9387	0.9381	0.9284	0.9512	**0.9612**	0.9224	0.9462	0.9508
wine-1	0.9188	0.8193	**0.9971**	0.9966	0.9964	0.9752	0.9965	0.9319	0.9900	0.9970
wine-2	0.9014	0.8561	0.8920	0.9051	0.8908	**0.9368**	0.8880	0.9361	0.8533	0.8745
wine-3	0.9966	0.9479	0.9893	0.9913	0.9879	0.9906	0.9906	**0.9986**	0.9917	0.9891
sonar-rocks	0.6260	**0.6710**	0.6693	0.6000	0.5798	0.6613	0.6372	0.6213	0.6497	0.6289
sonar-mines	0.6694	0.6535	0.6730	0.6039	0.5718	0.5973	0.6710	0.6454	**0.7252**	0.6177
seed-kama	0.9410	0.9220	0.9566	**0.9598**	0.9594	0.9537	0.9467	0.8734	0.9138	0.9549
seed-rosa	0.9869	**0.9930**	0.9891	0.9884	0.9859	0.9884	0.9754	0.8751	0.9703	0.9882
seed-canadian	0.9817	**0.9831**	0.9737	0.9763	0.9730	0.9758	0.9664	0.9137	0.9763	0.9749
heart-absence	0.8173	0.7442	0.7790	0.8301	0.7918	**0.8641**	0.7942	0.7371	0.7453	0.8074
heart-presence	0.6631	0.6906	0.7288	0.7808	0.7340	**0.8117**	0.7323	0.6480	0.7026	0.7925
ionosphere-good	0.9576	0.9132	0.9457	0.9716	0.9488	0.9128	0.9641	**0.9792**	0.9706	0.9276
ionosphere-bad	0.2214	0.3419	0.2759	0.2804	0.3281	**0.3549**	0.2408	0.2388	0.2606	0.2956
balance-left	0.9305	0.9235	0.8837	0.9278	**0.9435**	0.9411	0.9108	0.6760	0.7919	0.8992
balance-middle	0.8378	**0.9496**	0.5861	0.6669	0.4644	0.5330	0.5554	0.9431	0.7830	0.6785
balance-right	0.9315	0.9266	0.8848	0.9295	**0.9450**	0.9427	0.9122	0.7364	0.8069	0.9010
breastcancer-benign	0.9880	0.9874	**0.9954**	0.9946	0.8589	0.9950	0.9880	0.9466	0.9864	0.9942
breastcancer-malignant	0.6526	0.6779	0.9395	0.7319	0.9090	**0.9640**	0.8634	0.3229	0.9351	0.9615
Average Rank	4.9	5.7	4.8	4	5.95	**3.7**	5.25	7.35	6.35	4.95

주요 고려사항 및 마무리

- **이상탐지 방법론의 실제 활용 관련 연구 분야**
  - 하이퍼파라미터 최적화 (Hyperparameter Optimization)
  - 이상탐지 앙상블 (Ensemble of Anomaly Detection Models)

- **이상탐지 관련 연구 분야**
  - 불확실성 정량화 (Uncertainty Quantification)
  - 학습 외 분포 탐지 (Out-Of-Distribution Detection)

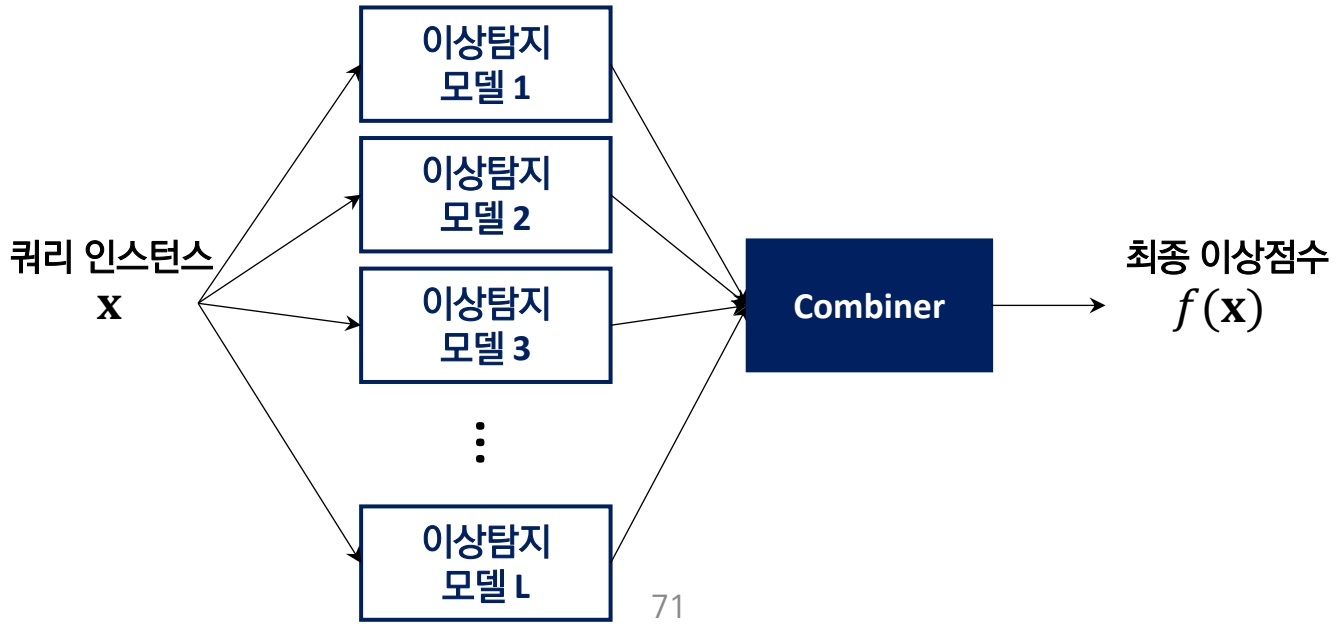
- Takeaway

하이퍼파라미터 최적화 (Hyperparameter Optimization)

- 학습 데이터셋 내 이상 인스턴스의 부재 상황에서 직접적으로 하이퍼파라미터를 최적화하는 것은 불가능하며 이를 위한 대안들이 제안되어 옴
  - 민감도가 낮은 이상탐지 모델 활용
    - ✓ Isolation Forest는 하이퍼파라미터 설정에 대한 민감도가 낮음
  - 모델 특화 (Model-Specific) 방법
    - ✓ Parzen Window: bandwidth optimization methods
    - ✓ OC-SVM: Quick Model Selection (QMS)
    - ✓ …
  - 모델 불특정 (Model-Agnostic) 방식
    - ✓ Synthetic Anomaly Generation: 인공 이상 데이터를 생성 후 이를 학습 데이터로부터 잘 분리하도록 이상탐지 모델의 하이퍼파라미터 최적화
    - ✓ Outlier Exposure: 학습 외 분포 특성을 갖는 외부 데이터셋을 활용
    - ✓ Clustering-based Proxy Measure (C-Proxy): 학습 데이터셋을 K개의 군집으로 분할한 후, 임의의 K-1개의 군집이 나머지 군집과 분리되도록 이상탐지 모델의 하이퍼파라미터 최적화

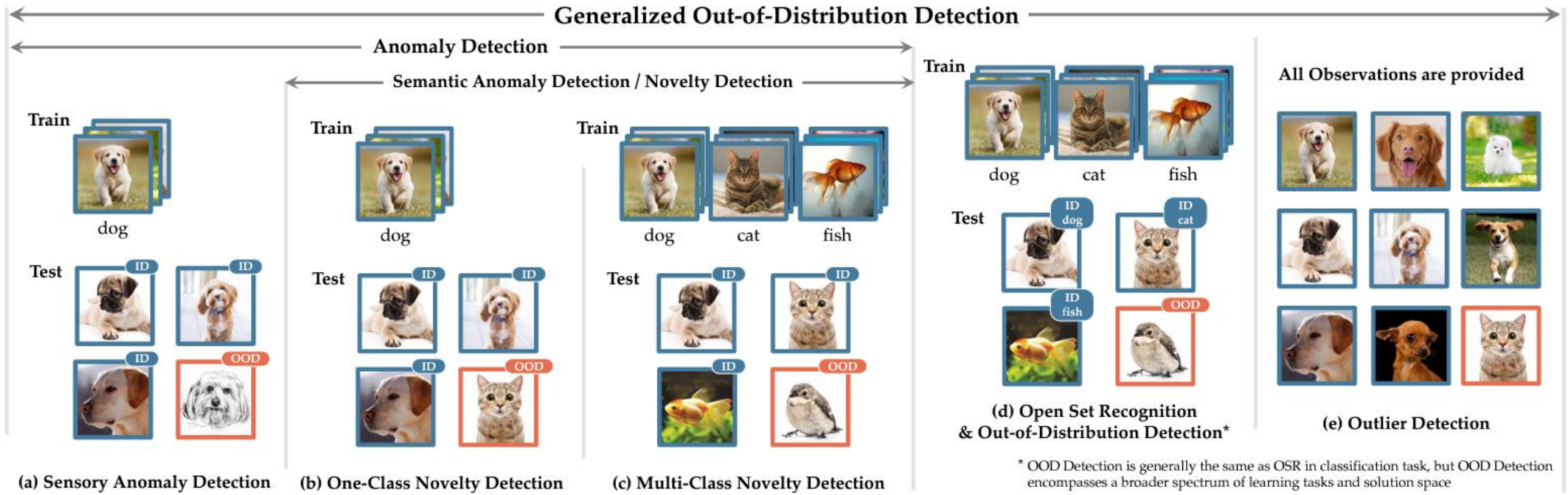
이상탐지 앙상블 (Ensemble of Anomaly Detection Models)

- **여러 개의 이상탐지 모델의 앙상블을 통한 이상탐지 성능 및 안정성 개선**
  - 다양한 이상탐지 방법론을 이용하여 여러 이상탐지 모델을 학습할 수 있으나, 이 중에서 어떤 모델이 가장 좋은 성능을 보이는지 사전에 알기 어려움
  - 주어진 쿼리 인스턴스에 대해서 여러 개의 이상탐지 모델이 도출하는 이상점수를 결합하여 최종 이상점수 도출
  - 예시: 개별 이상탐지 모델로부터 얻은 이상점수의 평균이 높으면 이상으로 탐지, 하나의 이상탐지 모델이라도 이상점수가 높으면 이상으로 탐지
  - **주의사항:** 개별 이상탐지 모델의 이상점수 스케일이 다른 경우 정규화 필요



학습 외 분포 탐지 (Out-Of-Distribution Detection)

- 넓은 의미로는, 모델이 이전에 학습한 데이터의 분포에서 벗어나는 인스턴스를 탐지하는 연구 분야에 대한 일반화된 개념
  - Open-Set Recognition & Out-Of-Distribution Detection: 분류 문제에 대해서 모델이 학습하지 않은 새로운 범주의 인스턴스가 등장할 때 이를 탐지하는 방법



Takeaway

- **지도학습 기반 이상탐지의 한계**
  - **학습 데이터에 이상 인스턴스를 충분히 많이/다양하게 확보할 수 있으면 최선의 접근**
  - 그러나, 모든 발생 가능한 이상 유형이 학습 데이터셋에 존재하지 않을 가능성이 큼
  - 학습 데이터셋 내 이상 데이터의 양이 매우 적은 극심한 범주 불균형 상황에 적절하지 않음
  - 불확실성 정량화 & 학습 외 분포 탐지 기법을 활용하여 한계점을 보완 가능

- **비지도학습 기반 이상탐지의 한계**
  - **학습 데이터에 이상 인스턴스가 부재한 상황에서 활용할 수 있는 접근**
  - 실제 이상 데이터의 부재 상황에서 구축한 이상탐지 모델이 잘 작동하는지에 대한 정량적 평가가 불가능
  - 그러나, 이상탐지 모델의 평가를 위한 노력이 없이는 실제 활용이 무의미해질 수 있음
  - XAI기법을 활용한 이상탐지 모델의 설명이 모델 평가의 실마리가 될 수 있음

Takeaway

- **머신러닝 기반 이상탐지의 실효성 측면**
  - 명확하게 드러나는 이상을 탐지하는데 매우 효과적 (규칙기반으로도 쉽게 탐지할 수 있음)
  - 이상과 노이즈 간 구분이 잘 되지 않는 경우, 이상을 잘 탐지하지 못함

- **No-Free-Lunch**
  - 다양한 이상탐지 방법론 중 내가 가지고 있는 문제에 최선의 방법론이 무엇인지 사전에 알 수 없음
  - 여러 방법론의 적용을 시도하고 결과를 비교하는 과정이 필요함

- **이상탐지 모델의 활용**
  - 이상탐지 모델은 과거 학습 데이터의 표현과 다른 새로운 데이터 표현의 인스턴스들을 이상으로 판단하며, 데이터 표현 측면의 이상이 항상 실제 도메인 관점에서의 이상을 의미하지는 않음
  - 데이터 표현 측면의 이상이 도메인 관점에서의 이상과 일치하도록 하는 데이터 표현을 도출하는 과정이 매우 중요

감사합니다.

문의사항 E-mail: s.kang@skku.edu