기초 이론부터 실무 실습까지

머신 러닝 익히기

programmers

Part 08. 이상 탐지

정 정 민

programmers

Chapter 18. Isolation Forest (고립 숲)

1. 이상 탐지 문제
2. Isolation Forest (고립 숲)

이상 탐지 문제

[RECAP] 이상 탐지, Anomaly Detection

- 데이터에서 비정상적인 패턴, 이상치, 또는 예외적인 사례를 탐지하는 과정

- 데이터에서 일반적으로 볼 수 있는 특성에서 많이 벗어난 데이터를 식별하는 과정에서 사용
- 보안, 금융, 의료 등의 분야에서 중요한 역할

- 예를 들어,
  - 나의 계좌가 갑자기 외국 어딘가에서 로그인 하려는 시도가 포착되었거나
  - 충격파 그래프를 이용해 물체 혹은 건물 내부의 균열을 찾아낸다거나

전통적 이상 탐지 방법

- 이전에 진행한 실습 과정에서도 이상치를 판단하고 제거 했음

- 전통적인 통계 기반 방법으로 대표적으로
  - **IQR (Interquartile Range)**
    - 1사분위수(Q1)와 3사분위수(Q3)간의 차이로
    - Q1-1.5IQR 미만 혹은 Q3+1.5IQR 초과 데이터를 이상치로 간주
  - **Z-Score**
    - 데이터 포인트가 평균으로부터 표준 편차의 몇 배만큼 떨어져 있는지를 나타내는 수치
    - 이 수치가 3이상인 데이터를 이상치로 간주

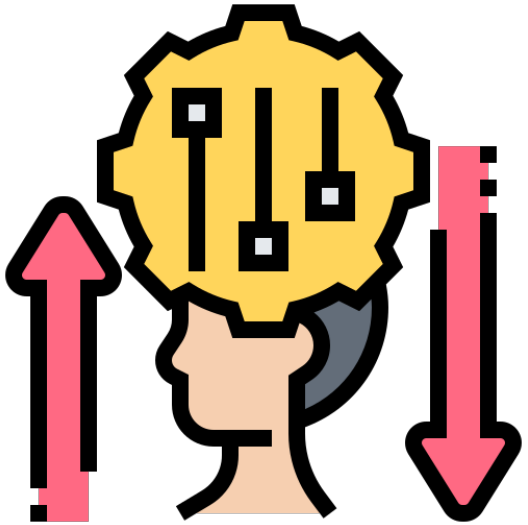
- 계산이 복잡하지 않고 해석이 직관적
- 하지만, 모든 데이터에 적용해 의미있는 결과를 얻기에는 어렵고
- 데이터가 대칭이 아니거나 복잡한 분포를 갖고 있다면 오인식이 늘어남

머신 러닝을 활용한 이상 탐지

- 고차원 & 대량의 데이터는 그 내부의 패턴을 갖고 있음

- 머신 러닝 모델로 데이터의 패턴을 익히고
- 그 패턴에서 벗어난 데이터를 이상치로 취급

- 따라서
  - 복잡한 데이터의 패턴에 적응적이며
  - 지속적인 학습과 개선이 가능
- 하지만
  - 처리할 모델이 복잡할 수 있고
  - 오히려 사람이 해석하기 어려움 패턴이 존재할 수 있음
  - 그리고 양질의 데이터가 다수 필요함



예를 들어,

- **매년 1회 외국으로 여행을 가는 사람의 소비 패턴**
  - 전통적인 방법은 모든 외국 소비 패턴을 이상치로 취급할 것
  - 머신 러닝 방법은 외국 여행의 패턴을 익히고 이상치로 취급하지 않을 것

- **산업 공정 내 불량품 찾기**
  - 전통 방법 : 임계치로 이상 탐지를 할 수 밖에 없음
  - 머신 러닝 : 시간에 따른 온도, 압력 등의 변화까지 모두 고려 가능

- **혼수 준비**
  - 전통 방법 : 갑작스러운 소비 증가로 이상치 취급
  - 머신 러닝 : 비슷한 인구 통계학적 사람들이 보이는 특성을 학습해 이상치로 보지 않을 수 있음

- 등등

Isolation Forest

강의 실습 코드 : 링크

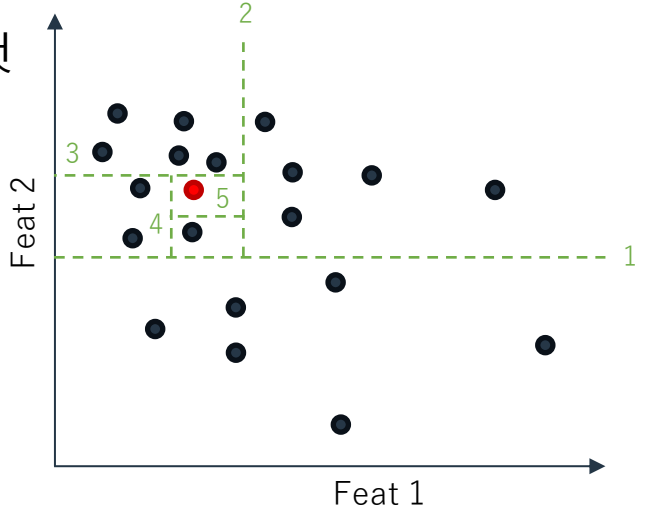
개념 잡기

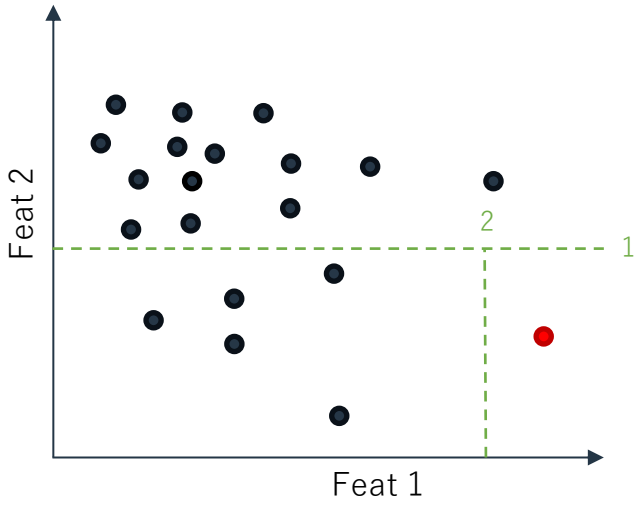
- 정상 데이터는 그와 비슷한 특성 패턴을 갖는 비슷한 데이터와 밀도 있게 모여있을 것
- 반면, 이상치 데이터는 밀도가 낮은 공간에 존재

- 이때, 하나의 데이터를 '고립'하도록 어떤 특성과 그것의 분할 값을 기준으로 나눔
  - 마치 Decision Tree 처럼..

- 이때 정상 데이터는
  - 밀도가 높은 구역에 있으니
  - 많은 분할 과정이 지나야 '고립'되며

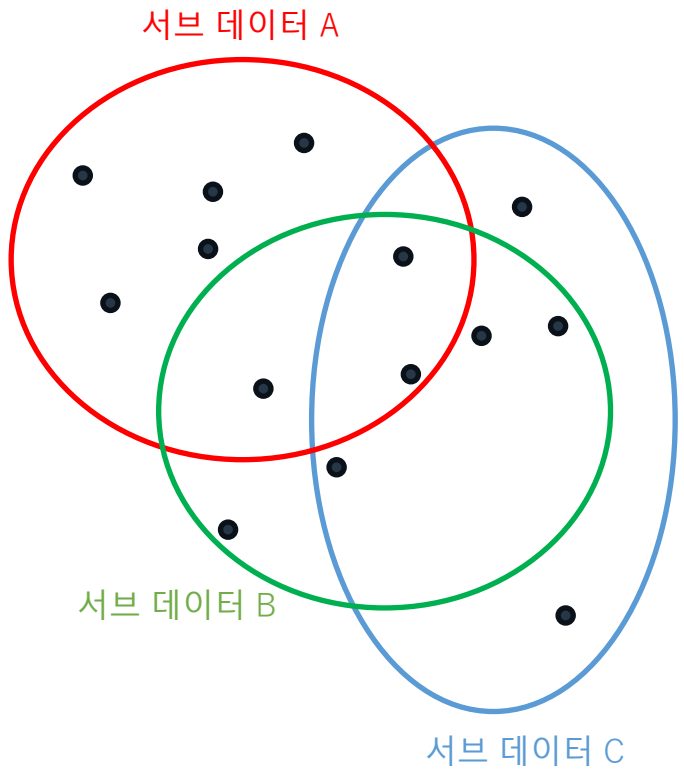
- 이상치 데이터는
  - 밀도가 낮은 지역에 있어서
  - 낮은 수준의 분할 과정으로도 쉽게 고립시킬 수 있음





작동 과정

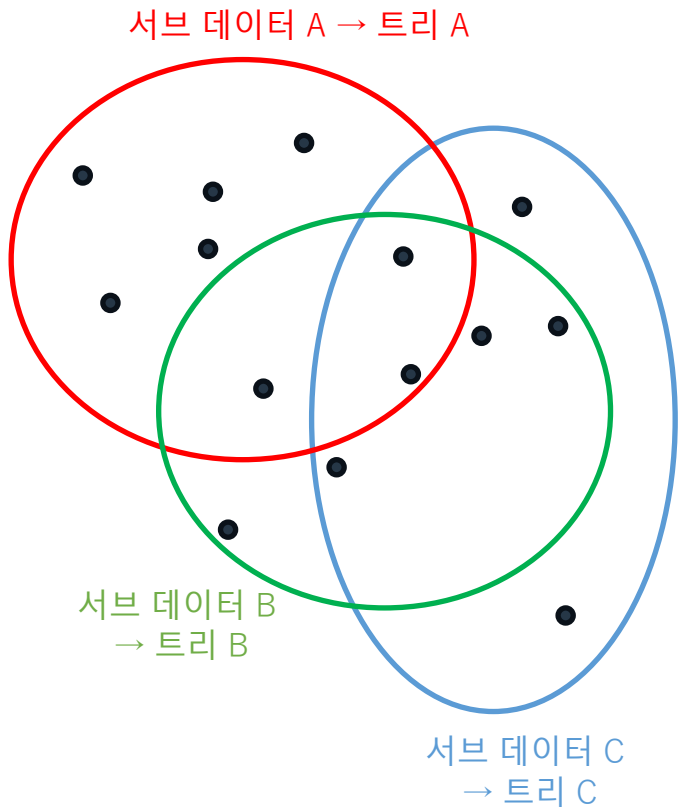
- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨
- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음



작동 과정

- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨

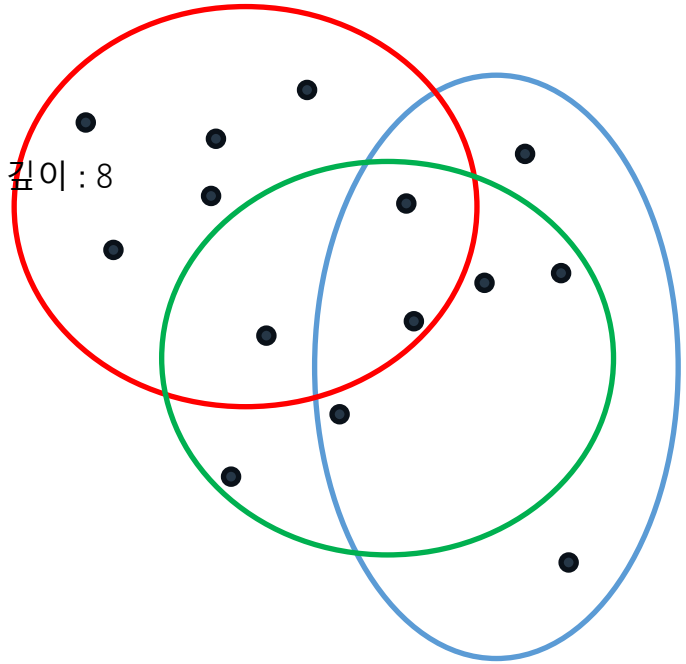
- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음
- **트리 생성** : 하나의 서브 데이터셋 마다 개별 Isolation Tree(고립 트리)를 준비



작동 과정

- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨

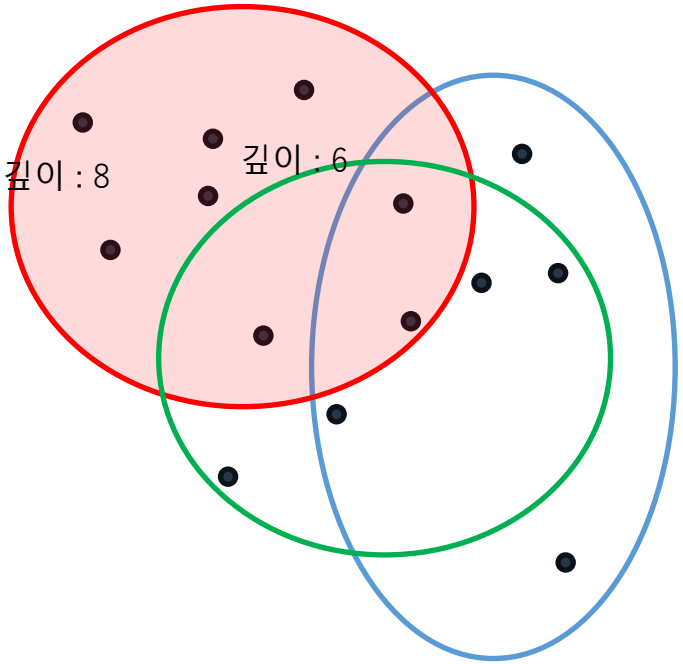
- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음
- **트리 생성** : 하나의 서브 데이터셋 마다 개별 Isolation Tree(고립 트리)를 준비
- **고립 시작** : 각 트리에 소속된 데이터셋을 하나씩 고립
  - 임의의 특성과 임의의 특성 분할 값을 선택
  - 데이터셋이 리프 노드에 도달할 때까지 반복
  - 각 데이터 포인트는 리프 노드까지 도달한 깊이를 저장하고 있음



작동 과정

- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨

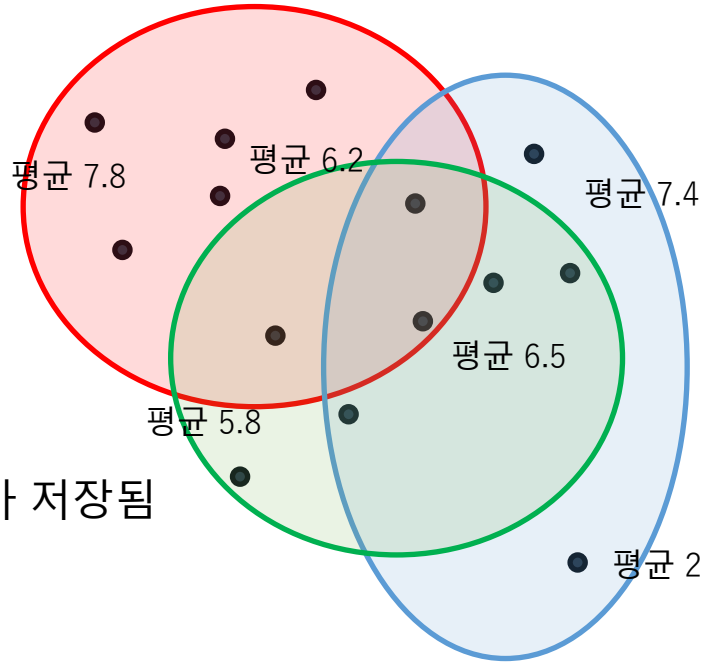
- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음
- **트리 생성** : 하나의 서브 데이터셋 마다 개별 Isolation Tree(고립 트리)를 준비
- **고립 시작** : 각 트리에 소속된 데이터셋을 하나씩 고립
  - 임의의 특성과 임의의 특성 분할 값을 선택
  - 데이터셋이 리프 노드에 도달할 때까지 반복
  - 각 데이터 포인트는 리프 노드까지 도달한 깊이를 저장하고 있음
- **고립 완료** : 하나의 트리에 소속된 모든 데이터셋을 전부 고립



작동 과정

- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨

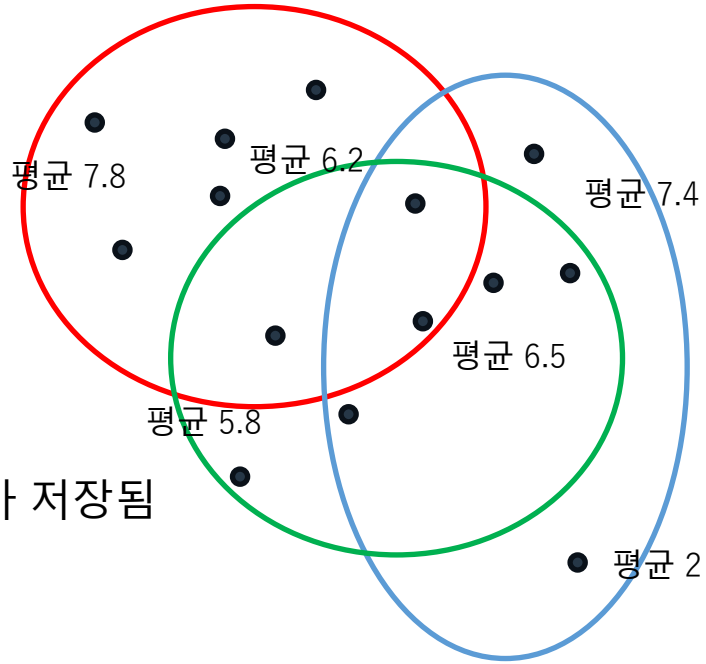
- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음
- **트리 생성** : 하나의 서브 데이터셋 마다 개별 Isolation Tree(고립 트리)를 준비
- **고립 시작** : 각 트리에 소속된 데이터셋을 하나씩 고립
  - 임의의 특성과 임의의 특성 분할 값을 선택
  - 데이터셋이 리프 노드에 도달할 때까지 반복
  - 각 데이터 포인트는 리프 노드까지 도달한 깊이를 저장하고 있음
- **고립 완료** : 하나의 트리에 소속된 모든 데이터셋을 전부 고립
- **트리들 계산 완료** : 모든 트리에서 고립 과정 완료
  - 여러 트리에 소속된 어떤 데이터 포인트는 서로 다른 트리에서 계산된 깊이가 저장됨



작동 과정

- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨

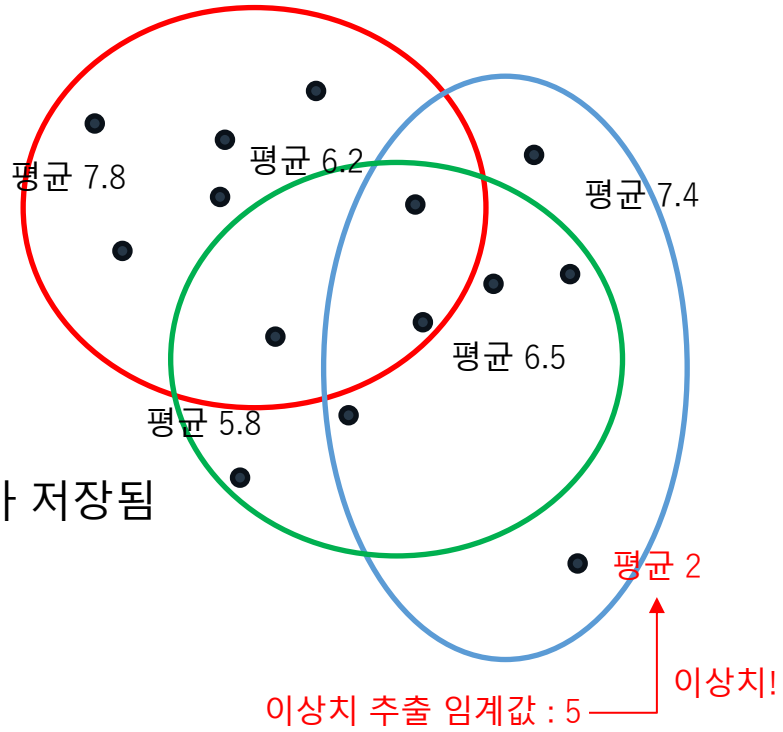
- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음
- **트리 생성** : 하나의 서브 데이터셋 마다 개별 Isolation Tree(고립 트리)를 준비
- **고립 시작** : 각 트리에 소속된 데이터셋을 하나씩 고립
  - 임의의 특성과 임의의 특성 분할 값을 선택
  - 데이터셋이 리프 노드에 도달할 때까지 반복
  - 각 데이터 포인트는 리프 노드까지 도달한 깊이를 저장하고 있음
- **고립 완료** : 하나의 트리에 소속된 모든 데이터셋을 전부 고립
- **트리들 계산 완료** : 모든 트리에서 고립 과정 완료
  - 여러 트리에 소속된 어떤 데이터 포인트는 서로 다른 트리에서 계산된 깊이가 저장됨
- **이상치 점수 계산** : 평균 깊이를 바탕으로 각 데이터 포인트의 이상치 점수를 계산



작동 과정

- 아래의 과정을 통해 Isolation Forest 알고리즘이 작동 됨

- **데이터 준비** : 전체 데이터에서 무작위로 서브 데이터셋을 선택
  - 어떤 데이터는 여러 서브 데이터셋에 속할 수 있음
- **트리 생성** : 하나의 서브 데이터셋 마다 개별 Isolation Tree(고립 트리)를 준비
- **고립 시작** : 각 트리에 소속된 데이터셋을 하나씩 고립
  - 임의의 특성과 임의의 특성 분할 값을 선택
  - 데이터셋이 리프 노드에 도달할 때까지 반복
  - 각 데이터 포인트는 리프 노드까지 도달한 깊이를 저장하고 있음
- **고립 완료** : 하나의 트리에 소속된 모든 데이터셋을 전부 고립
- **트리들 계산 완료** : 모든 트리에서 고립 과정 완료
  - 여러 트리에 소속된 어떤 데이터 포인트는 서로 다른 트리에서 계산된 깊이가 저장됨
- **이상치 점수 계산** : 평균 깊이를 바탕으로 각 데이터 포인트의 이상치 점수를 계산
- **이상치 추출** : 이상치 점수와 특정 임계값을 비교해 이상치 추출



Isolation Forest 활용해 학습하기

- n_estimatores
  - Isolation Forest를 구성하는 고립 트리의 수
  - 서브 데이터셋의 수와 같음
  - 더 많은 트리를 사용할수록 이상치 탐지의 안정성과 정확성이 향상
  - 하지만, 트리 수가 많을수록 계산 시간과 메모리 사용량도 증가
- max_samples
  - 서브 데이터셋에 포함될 최대 데이터 포인트의 수
  - 너무 적은 샘플은 모델의 성능을 저하시킬 수 있음
  - 너무 많은 샘플은 계산 비용을 증가시킴
- contamination
  - 전체 데이터셋에서 이상치 데이터가 차지할 비율
  - 이 비율을 바탕으로 정상 데이터와 이상치 데이터 사이의 이상치 점수 임계값을 설정할 수 있음
  - 단, 이는 데이터 내의 이상치가 어느 정도 있음을 알고 있는 상황에서 유용함
  - 만약 이상치의 정확한 비율을 모르거나 데이터가 복잡하다면 'auto'를 사용
    - 데이터 분포 특성에 따라 합리적 비율을 자동 추출

```
from sklearn.ensemble import IsolationForest
IForest = IsolationForest(n_estimators=100,
                          max_samples='auto',
                          contamination='auto')
IForest.fit(X)
```

이상치 탐지 평가 과정

- 이상치 데이터 정답을 알고 있다면
- 이상 데이터를 양성으로 놓고 분류 형태의 문제로 평가 가능

- 분류 문제에서 흔히 사용하는 수치들을 사용 사용 가능
  - 정확도(Accuracy)
  - 정밀도(Precision)
  - 재현율(Recall)
  - F1 값

- 만약 정답이 없다면
  - 시각화를 통한 전문가 검토를 진행

```
from sklearn.metrics import accuracy_score,
                            precision_score,
                            recall_score,
                            f1_score

accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
```

E.O.D