이상 탐지 (Anomaly Detection)

- 데이터셋에서 다른 대부분의 데이터들과 확연히 다른 패턴을 보이는 데이터, 즉 **이상치(Anomaly)** 또는 **특이치(Outlier)**를 식별하는 비지도 학습 기법
- 시스템 오류, 사기 거래, 희귀 사건, 측정 에러 등 다양한 원인으로 발생 가능
- 이를 찾아내는 것은 데이터 분석의 품질을 높이고 중요한 인사이트를 발견하는 데 매우 중요

주요 이상 탐지 알고리즘 종류

1. **Isolation Forest (고립 포레스트)**:
    - **개념**: 앙상블 기반의 이상 탐지 알고리즘으로, "이상치는 정상 데이터보다 더 쉽게 고립(분리)될 수 있다"는 아이디어에 기반합니다.
    - **동작 방식**:
        1. 데이터셋에서 무작위로 샘플을 선택하고, 선택된 샘플 내에서 무작위로 특성과 분할 지점을 골라 데이터를 분할하는 결정 트리(Isolation Tree)를 여러 개 생성합니다.
        2. 정상 데이터는 고립시키기 위해 많은 분할이 필요한 반면, 이상치는 트리의 상단부(root에 가까운)에서 적은 횟수의 분할만으로도 쉽게 고립됩니다.
        3. 각 데이터 포인트가 모든 트리에서 고립되기까지 필요한 평균 경로 길이(path length)를 계산합니다.
        4. 평균 경로 길이가 짧을수록 이상치일 가능성이 높다고 판단하여 '이상 점수(anomaly score)'를 부여합니다.
    - **장점**: 대용량 데이터와 고차원 데이터에서도 빠르고 효율적으로 작동합니다. 데이터 분포에 대한 가정이 필요 없습니다.
    - **단점**: 데이터셋 내에 이상치가 너무 많으면(e.g., >10%) 탐지 성능이 저하될 수 있습니다.
2. **Local Outlier Factor (LOF)**:
    - **개념**: 데이터의 **지역적 밀도(Local Density)**를 기반으로 이상치를 탐지하는 알고리즘입니다. "이상치는 주변 데이터들과 비교했을 때 밀도가 현저히 낮다"는 아이디어를 사용합니다.
    - **동작 방식**:
        1. 각 데이터 포인트에 대해, 가장 가까운 K개의 이웃(k-neighbors)까지의 거리를 계산합니다.
        2. 이를 바탕으로 각 데이터 포인트의 '지역적 도달 가능성 밀도(local reachability density)'를 계산합니다.
        3. 각 데이터 포인트의 밀도를 그 이웃들의 평균적인 밀도와 비교하여 'Local Outlier Factor(LOF)' 점수를 계산합니다.
        4. LOF 점수가 1보다 현저히 크면, 해당 포인트는 주변 이웃들보다 밀도가 훨씬 낮다는 의미이므로 이상치로 간주합니다.
    - **장점**: 데이터의 전체적인 분포가 아닌 지역적인 밀도를 고려하므로, 여러 군집이 서로 다른 밀도를 가질 때도 효과적으로 이상치를 탐지할 수 있습니다.
    - **단점**: 모든 데이터 포인트에 대해 이웃과의 거리를 계산해야 하므로 계산 비용이 높고, 대용량 데이터에는 부적합합니다. 고차원 데이터에서는 거리 계산의 어려움으로 성능이 저하될 수 있습니다.

적용 가능한 상황

- 금융 거래에서의 사기 탐지 (Fraud Detection).

- 네트워크 침입 탐지, 시스템 오류 및 비정상 행위 감지.
- 제조 공정에서의 불량품 검출.
- 의료 데이터에서 희귀 질병이나 비정상적인 생체 신호 발견.

구현 방법

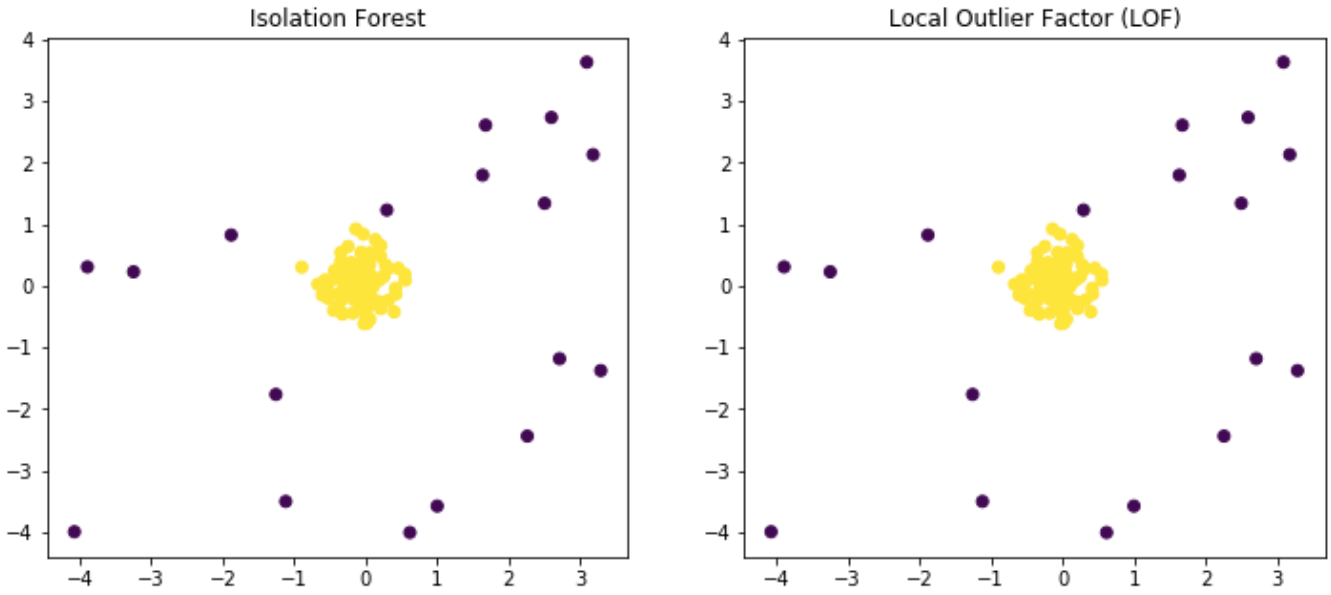
`scikit-learn`의 `ensemble` 모듈에 있는 `IsolationForest`와 `neighbors` 모듈의 `LocalOutlierFactor`를 사용합니다.

주의사항

- `contamination` **파라미터**:
  - 두 알고리즘 모두 `contamination`이라는 하이퍼파라미터를 보유
  - 데이터셋에 포함된 이상치의 비율을 의미
  - 이 값을 어떻게 설정하느냐에 따라 이상치를 판별하는 임계값이 결정
  - 기본값은 'auto' 또는 0.1
- **특성 스케일링**:
  - Isolation Forest는 트리 기반이므로 필수는 아니지만, 일관성을 위해 스케일링 적용을 추천
  - LOF는 거리 기반 알고리즘이므로 스케일링이 중요

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import StandardScaler

# 1. 데이터 생성 (정상 데이터와 이상치)
np.random.seed(42)
X_normal = 0.3 * np.random.randn(100, 2)
X_outliers = np.random.uniform(low=-4, high=4, size=(20, 2))
X = np.r_[X_normal, X_outliers]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. Isolation Forest
# 주요 하이퍼파라미터
# n_estimators: 생성할 트리의 개수.
# contamination: 데이터 내 이상치의 비율.
# max_samples: 각 트리를 훈련할 때 사용할 샘플의 비율 또는 개수.
iso_forest = IsolationForest(n_estimators=100, contamination=0.15,
random_state=42)
y_pred_iso = iso_forest.fit_predict(X_scaled) # 예측 결과: 1(정상), -1(이상치)

# 3. Local Outlier Factor (LOF)
# 주요 하이퍼파라미터
# n_neighbors: 이웃의 수 (K).
# contamination: 데이터 내 이상치의 비율.
# novelty=True로 설정하면, fit()으로 학습 후 새로운 데이터에 대해 predict() 가능
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.15)
```

```
y_pred_lof = lof.fit_predict(X_scaled) # 예측 결과: 1(정상), -1(이상치)

# 4. 결과 시각화
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_pred_iso, cmap='viridis')
plt.title('Isolation Forest')

plt.subplot(1, 2, 2)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_pred_lof, cmap='viridis')
plt.title('Local Outlier Factor (LOF)')

plt.show()

print("Isolation Forest가 탐지한 이상치 개수:", np.sum(y_pred_iso == -1))   # 18
print("LOF가 탐지한 이상치 개수:", np.sum(y_pred_lof == -1))               # 18
```



장단점 및 대안

- **Isolation Forest**:
  - **장점**: 빠르고, 대용량/고차원 데이터에 확장성이 좋습니다.
  - **단점**: 군집 내부에 위치한 이상치나, 데이터 밀도가 다양한 경우 탐지 성능이 떨어질 수 있습니다.
- **Local Outlier Factor (LOF)**:
  - **장점**: 데이터의 지역적 밀도를 고려하므로 다양한 밀도를 가진 데이터셋에서도 잘 작동합니다.
  - **단점**: 계산 비용이 높아 대용량 데이터에 적용하기 어렵습니다.
- **대안**:
  - **One-Class SVM**: 정상 데이터만을 학습하여, 정상 데이터의 경계를 만들고 그 경계에서 벗어나는 데이터를 이상치로 탐지하는 기법입니다.
  - **DBSCAN**: 밀도 기반 군집 알고리즘으로, 어떤 군집에도 속하지 않는 데이터를 노이즈(이상치)로 자연스럽게 식별할 수 있습니다.