강건 회귀 (Robust Regression)

- 데이터에 포함된 이상치(outlier)의 영향을 줄여, 보다 안정적이고 신뢰성 있는 회귀 모델을 구축하기 위한 기법
- 일반적인 최소제곱법(OLS) 기반의 선형 회귀는 이상치에 매우 민감하여 회귀선이 왜곡될 수 있음
- 강건 회귀는 이상치에 가중치를 적게 부여하거나 다른 손실 함수를 사용하여 이상치에 민감한 문제를 완화

적용 가능한 상황

- 데이터에 이상치가 존재할 것으로 의심되거나, 실제로 확인되었을 때 사용합니다.
- 이상치로 인해 기존 선형 회귀 모델의 성능이 저하되고 예측이 불안정할 때 적용합니다.
- 데이터 정제 과정에서 이상치를 제거하기 곤란하거나, 이상치 자체도 중요한 정보를 담고 있을 수 있어 무작정 제거하고 싶지 않을 때 유용합니다.

구현 방법

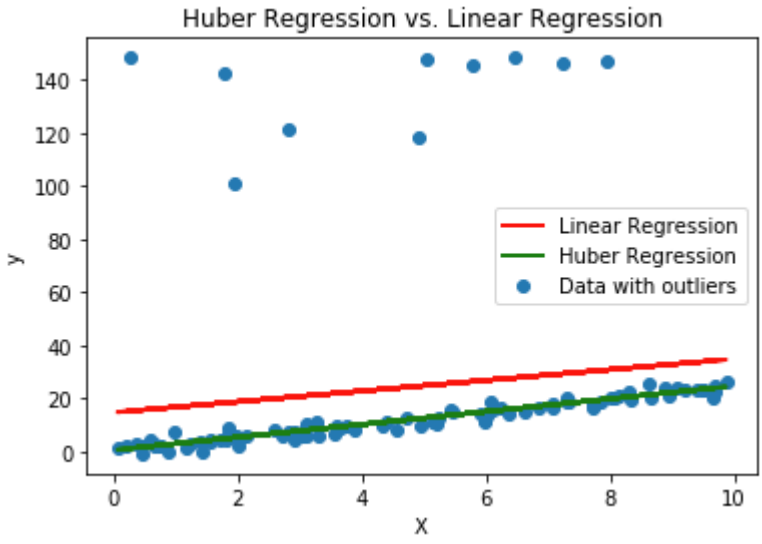
강건 회귀는 `statsmodels`와 `scikit-learn` 라이브러리를 통해 구현할 수 있습니다. 대표적인 강건 회귀 모델로는 RANSAC, Theil-Sen, Huber 회귀 등이 있습니다.

1. Huber Regression

- **용도**
  - 오차의 크기가 특정 임계값(epsilon) 이내면, 제곱 오차(L2 손실)를 사용
  - 오차의 크기가 특정 임계값(epsilon) 벗어나면, 선형 오차(L1 손실)를 사용
  - L1과 L2 손실의 장점을 결합하여 이상치의 영향을 줄이는 형태
- **주의사항**
  - 하이퍼파라미터 `epsilon` 값에 따라 모델의 강건성이 달라짐
  - `epsilon`이 클수록 일반적인 선형 회귀(OLS)와 유사해지고, 작을수록 이상치에 더 강건

```
import numpy as np
import pandas as pd
from sklearn.linear_model import HuberRegressor, LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# 1. 데이터 생성 (이상치 포함)
np.random.seed(42)
X = np.random.rand(100, 1) * 10
y = 2.5 * X.ravel() + np.random.randn(100) * 2

# 이상치 추가
X[90:] = np.random.rand(10, 1) * 10
y[90:] = np.random.rand(10) * 50 + 100

# 2. 모델 학습
# HuberRegressor 하이퍼파라미터
# epsilon: L1과 L2 손실을 전환하는 임계값. 이상치로 판단할 기준을 정합니다. (기본값
```

```
=1.35)
#   - 데이터의 스케일에 따라 조정이 필요할 수 있습니다.
# alpha: 규제 강도. 모델의 복잡도를 제어합니다. (기본값=0.0001)
huber = HuberRegressor(epsilon=1.35)
huber.fit(X, y)

# 비교를 위한 일반 선형 회귀
lr = LinearRegression()
lr.fit(X, y)

# 3. 결과 시각화
plt.scatter(X, y, label='Data with outliers')
plt.plot(X, lr.predict(X), color='red', linewidth=2, label='Linear Regression')
plt.plot(X, huber.predict(X), color='green', linewidth=2, label='Huber
Regression')
plt.title('Huber Regression vs. Linear Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()

print(f"Huber 회귀 계수: {huber.coef_}, 절편: {huber.intercept_}")
print(f"선형 회귀 계수: {lr.coef_}, 절편: {lr.intercept_}")
'''
Huber 회귀 계수: [2.41020091], 절편: 0.6654725461601323
선형 회귀 계수: [2.01154445], 절편: 14.826638951529306
'''
```



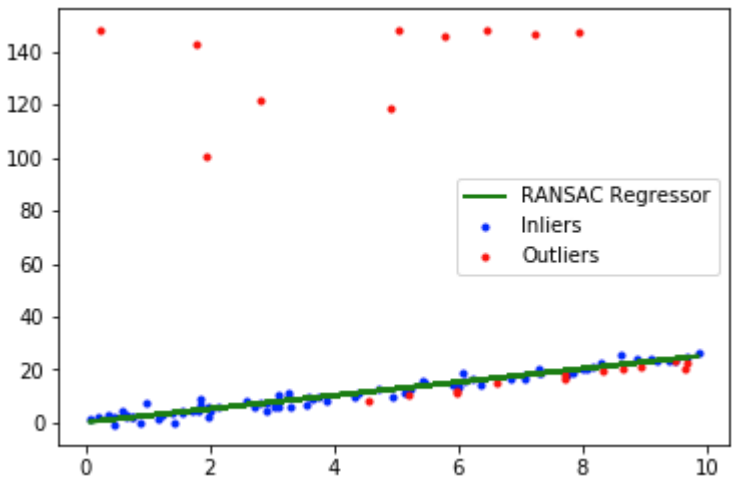
- **결과 해석**: 시각화된 그래프에서 일반 선형 회귀선은 이상치 쪽으로 크게 기울어진 반면, Huber 회귀선은 이상치의 영향을 덜 받아 전체 데이터의 추세를 더 잘 따르는 것을 볼 수 있습니다. 회귀 계수를 비교해보면 Huber 회귀가 이상치의 영향을 덜 받은 모델을 생성했음을 확인할 수 있습니다.

2. RANSAC (RANdom SAmple Consensus)

- **용도**

    - 전체 데이터 중 일부를 무작위로 샘플링하여 '정상치(inlier)'로 간주하고, 이 정상치만으로 모델을 학습하는 과정을 반복
    - 가장 많은 정상치를 포함하는 최적의 모델을 최종 모델로 선택
- **주의사항**
    - 정상치의 비율을 어느 정도 예측해야 함
    - 무작위 샘플링으로 인해 실행할 때마다 결과가 약간씩 달라질 수 있습니다.

```
from sklearn.linear_model import RANSACRegressor

# RANSACRegressor 하이퍼파라미터
# base_estimator: 사용할 기본 모델 (e.g., LinearRegression). (기본값
=LinearRegression())
# min_samples: 무작위로 선택할 최소 샘플의 개수. (기본값=None, X.shape[1] + 1)
# residual_threshold: 정상치로 판단할 최대 잔차(오차)의 크기. (기본값=MAD of y)
# max_trials: 최대 반복 횟수. (기본값=100)
ransac = RANSACRegressor(base_estimator=LinearRegression(), min_samples=50,
residual_threshold=5.0, random_state=42)
ransac.fit(X, y)

# 결과 시각화
inlier_mask = ransac.inlier_mask_
outlier_mask = np.logical_not(inlier_mask)

plt.scatter(X[inlier_mask], y[inlier_mask], color='blue', marker='.',
label='Inliers')
plt.scatter(X[outlier_mask], y[outlier_mask], color='red', marker='.',
label='Outliers')
plt.plot(X, ransac.predict(X), color='green', linewidth=2, label='RANSAC
Regressor')
plt.legend()
plt.show()

print(f"RANSAC 회귀 계수: {ransac.estimator_.coef_}, 절편:
{ransac.estimator_.intercept_}")
# RANSAC 회귀 계수: [2.52161901], 절편: 0.23172065055441493
```

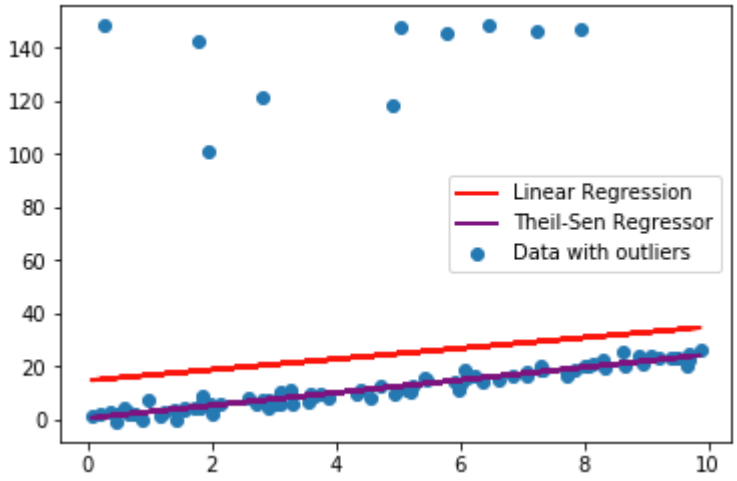


- **결과 해석**: RANSAC은 정상치(Inliers)와 이상치(Outliers)를 명시적으로 구분해줍니다. 시각화 결과에서 이상치로 탐지된 데이터 포인트를 확인할 수 있으며, 회귀선은 정상치 데이터만을 기반으로 학습되어 이상치의 영향을 받지 않은 것을 볼 수 있습니다.

3. Theil-Sen Regressor

- **용도**
  - 데이터의 모든 점 쌍(pair)에 대해 기울기를 계산한 후, 이 기울기들의 중앙값(median)을 최종 기울기로 선택
  - 절편 또한 중앙값을 사용하여 계산하므로 이상치에 매우 강건
- **주의사항**: 계산 복잡도가 높아($O(N^2)$) 대용량 데이터에는 적용하기 어려울 수 있습니다.

```
from sklearn.linear_model import TheilSenRegressor

# TheilSenRegressor 하이퍼파라미터
# n_subsamples: 기울기 계산 시 사용할 하위 샘플의 개수. 데이터가 클 때 계산 속도를 높이기 위해 사용. (기본값=n_samples)
# max_iter: 최대 반복 횟수. (기본값=300)
# random_state: 재현성을 위한 시드.
theil_sen = TheilSenRegressor(random_state=42)
theil_sen.fit(X, y)

# 결과 시각화
plt.scatter(X, y, label='Data with outliers')
plt.plot(X, lr.predict(X), color='red', linewidth=2, label='Linear Regression')
plt.plot(X, theil_sen.predict(X), color='purple', linewidth=2, label='Theil-Sen Regressor')
plt.legend()
plt.show()

print(f"Theil-Sen 회귀 계수: {theil_sen.coef_}, 절편: {theil_sen.intercept_}")
# Theil-Sen 회귀 계수: [2.38914674], 절편: 0.6428816171681937
```



- **결과 해석** Theil-Sen 회귀선 역시 이상치의 영향을 거의 받지 않고 데이터의 전반적인 추세를 잘 따릅니다. Huber 회귀보다도 이상치에 더 강건한 특성을 보일 때가 많습니다.

장단점 및 대안

- **장점**:
  - 이상치가 포함된 데이터에서도 안정적이고 신뢰도 높은 회귀 모델을 만들 수 있습니다.
  - 모델의 예측 성능 왜곡을 방지합니다.
- **단점**:
  - 일반 선형 회귀에 비해 계산 비용이 더 많이 듭니다.
  - 모델 종류에 따라 조정해야 할 하이퍼파라미터가 존재합니다.
  - 이상치가 없는 데이터에서는 일반 선형 회귀보다 성능이 약간 떨어질 수 있습니다.
- **대안**:
  - **이상치 제거/변환**: 강건 회귀를 사용하기 전에 데이터 전처리 단계에서 이상치를 직접 탐지하여 제거하거나, 다른 값(e.g., 평균, 중앙값)으로 대체하는 방법을 고려할 수 있습니다.
  - **분위수 회귀 (Quantile Regression)**: 데이터의 특정 분위수를 예측하므로 중앙값(50% 분위수) 회귀를 사용하면 이상치에 덜 민감한 모델을 얻을 수 있습니다.