선형 회귀 (Linear Regression)

- 지도 학습의 가장 기본적인 회귀 알고리즘 중 하나로, 독립변수(특성)와 종속변수(타겟) 간의 선형 관계를 모델링
- 모델은 각 독립변수에 대한 가중치(회귀 계수)와 절편을 학습하며, 이들의 선형 결합을 통해 종속변수의 값을 예측
- 예측값과 실제값 사이의 평균 제곱 오차(Mean Squared Error, MSE)를 최소화하는 가중치와 절편을 찾는 것이 목표
- **모델 식**: $y_{pred} = w_1x_1 + w_2x_2 + ... + w_nx_n + b$
    - w_i: 각 특성(x_i)에 대한 가중치(회귀 계수)
    - b: 절편 (bias)
- **비용 함수 (Cost Function)**: $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_{pred_i})^2$

적용 가능한 상황

- 예측하고자 하는 종속변수가 연속적인 숫자일 때 사용
    - e.g. 주택 가격, 주가, 판매량 예측
- 변수들 간의 관계가 복잡하지 않고, 선형적이라고 가정할 수 있을 때 좋은 성능을 보임
- 다른 복잡한 모델을 시도하기 전, 성능의 기준점을 설정하기 위한 베이스라인 모델로 매우 유용

코드 예시

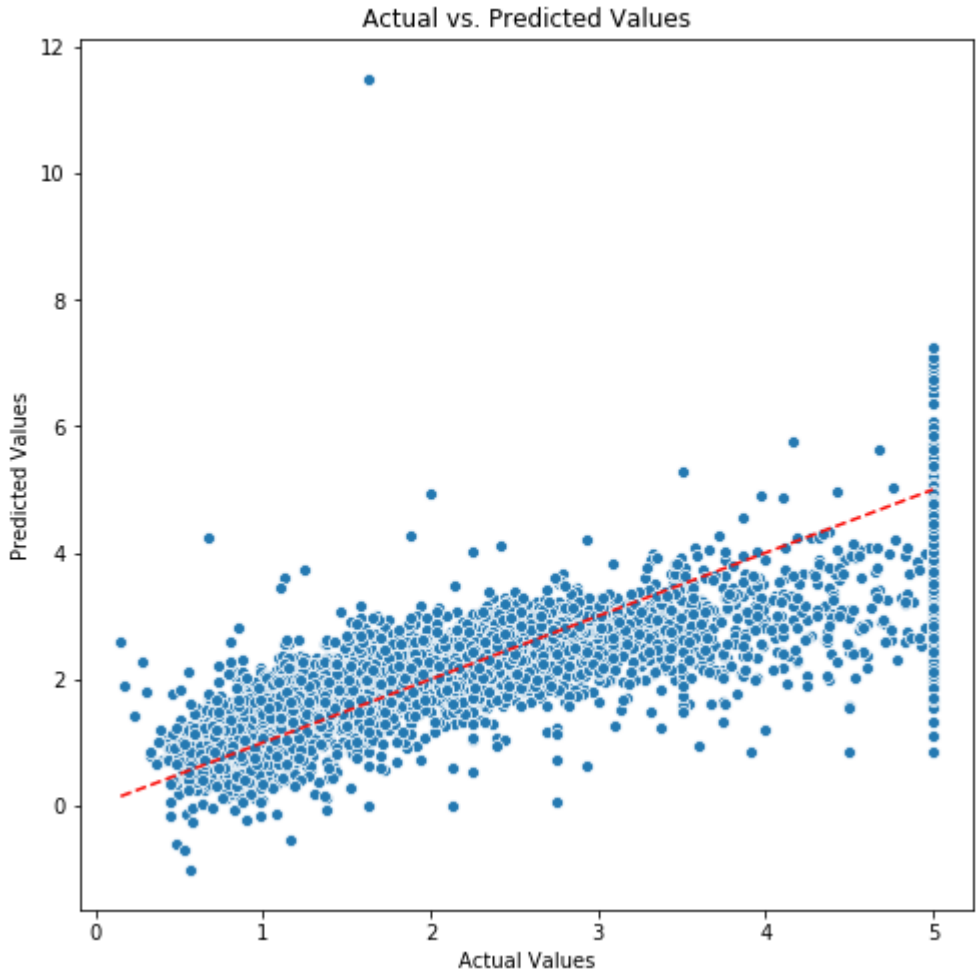
`scikit-learn`의 `LinearRegression`은 정규방정식(Normal Equation)을 사용하여 비용 함수를 최소화하는 계수를 해석적으로(analytically) 계산합니다.

주의사항

- **선형성 가정**
    - 특성과 타겟 간의 선형 관계를 가정하므로, 비선형 관계가 강한 데이터에서는 예측 성능 하락
- **이상치에 민감**
    - MSE를 비용 함수로 사용하므로, 오차가 큰 이상치에 민감하게 반응
        - 모델이 왜곡될 가능성이 높음
- **다중공선성**
    - 특성들 간에 강한 상관관계(다중공선성)가 존재하면 회귀 계수가 불안정해지고 해석이 어려워짐
    - 이 경우, 규제(Regularization)가 있는 릿지(Ridge)나 라쏘(Lasso) 회귀를 사용하는 것이 유용
- **특성 스케일링**
    - `LinearRegression`은 정규방정식을 사용하므로 특성 스케일링이 모델 성능에 직접적인 영향을 주지는 않음
    - 하지만, 경사 하강법 기반의 다른 회귀 모델(e.g., `SGDRegressor`)이나 규제 모델과 함께 사용할 때는 스케일링이 중요
    - 일관성을 위해 스케일링을 수행하는 것이 좋은 습관

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# 1. 데이터 준비
housing = fetch_california_housing()
X = pd.DataFrame(housing.data, columns=housing.feature_names)
y = housing.target

# 2. 데이터 분할 및 스케일링
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 3. 모델 학습
# LinearRegression 하이퍼파라미터
# fit_intercept: 절편(bias)을 계산할지 여부. (기본값=True)
# normalize: 데이터를 정규화할지 여부. StandardScaler를 사용하는 것이 권장되므로 False
로 둡니다. (Deprecated since version 1.0)
# n_jobs: 병렬 처리에 사용할 CPU 코어 수. -1이면 모든 코어 사용.
lin_reg = LinearRegression()
lin_reg.fit(X_train_scaled, y_train)

# 4. 예측 및 평가
y_pred = lin_reg.predict(X_test_scaled)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("--- 선형 회귀 모델 평가 ---")
print(f"MSE: {mse:.3f}")            # 0.556
print(f"RMSE: {rmse:.3f}")          # 0.746
print(f"R-squared (R²): {r2:.3f}")  # 0.576

# 5. 학습된 계수 확인
print("\n절편 (intercept):", lin_reg.intercept_) # 2.0719469373788777
coef_df = pd.DataFrame(lin_reg.coef_, index=X.columns, columns=['Coefficient'])
print("회귀 계수 (coefficients):\n", coef_df.sort_values(by='Coefficient',
ascending=False))
'''
             Coefficient
MedInc          0.854383
AveBedrms       0.339259
```

```
HouseAge       0.122546
Population    -0.002308
AveOccup      -0.040829
AveRooms      -0.294410
Longitude     -0.869842
Latitude      -0.896929
'''

# 6. 결과 시각화: 실제값 vs 예측값
plt.figure(figsize=(8, 8))
sns.scatterplot(x=y_test, y=y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--') # y=x 직선
plt.title('Actual vs. Predicted Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()
```



결과 해석 방법

- `intercept_`
  - 학습된 모델의 절편(bias) 값입니다.
  - 모든 특성 값이 0일 때의 예측값입니다.
- `coef_`: 각 특성에 대한 학습된 가중치(회귀 계수)입니다.

    - 계수의 부호(+/-): 해당 특성이 타겟 값에 긍정적인 영향(+)을 주는지, 부정적인 영향(-)을 주는지를 나타냅니다.
    - 계수의 크기: 해당 특성 값이 1단위 증가할 때 타겟 값이 얼마나 변하는지를 나타냅니다.
        - **단, 특성들이 동일한 스케일로 변환되었을 때만 크기를 직접 비교하여 영향력을 판단할 수 있습니다.**
- **평가 지표**:
    - **MSE/RMSE**: 예측 오차의 크기를 나타냅니다. 0에 가까울수록 좋은 모델입니다.
    - **R-squared (R^2)**: 모델이 데이터의 분산을 얼마나 잘 설명하는지를 나타내는 지표입니다. 1에 가까울수록 모델이 데이터를 잘 설명한다는 의미입니다.

장단점 및 대안

- **장점**:
    - 모델이 단순하여 학습 속도가 매우 빠르고, 대용량 데이터에도 적용 가능합니다.
    - 회귀 계수를 통해 각 특성이 결과에 미치는 영향을 직관적으로 해석할 수 있습니다.
- **단점**:
    - 데이터의 비선형 관계를 모델링하지 못합니다.
    - 이상치에 민감하고, 다중공선성 문제에 취약합니다.
- **대안**:
    - **다항 회귀**: 특성의 고차항을 추가하여 비선형 관계를 모델링합니다.
    - **규제 모델 (Ridge, Lasso, ElasticNet)**: 다중공선성 문제를 완화하고 모델의 과적합을 방지합니다.
    - **트리 기반 모델 (DecisionTree, RandomForest, GradientBoosting)**: 비선형성, 변수 간 상호작용을 잘 포착하며, 이상치에 상대적으로 덜 민감합니다.
    - **SVR (Support Vector Regressor)**: 커널 트릭을 사용하여 비선형 관계를 효과적으로 모델링할 수 있습니다。