2주차 (Chapter 3)

2주차 과제: 규제(Regularization) 강도에 따른 모델 계수 변화 분석 (상세 가이드)

1. 과제의 목표

이 과제의 목표는 선형 회귀 모델이 훈련 데이터에 과도하게 최적화되는 **과대적합(Overfitting)**을 어떻게 **규제(Regularization)**를 통해 완화할 수 있는지 이해하는 것입니다. 특히 규제의 강도를 조절하는 하이퍼파라미터인 `alpha` **값**을 바꿔가면서, 모델이 학습하는 **계수(coefficient)들의 크기**가 어떻게 변하는지 직접 확인하고 그 의미를 해석하는 데 중점을 둡니다.

2. 실험 과정 (Step-by-Step)

교재의 `03-3`절의 '릿지 회귀'와 '라쏘 회귀' 부분을 기반으로 진행합니다. 농어의 여러 특성을 사용해 무게를 예측하는 다중 회귀 예제입니다. 이 예제에서는 의도적으로 특성을 늘려 과대적합이 일어나기 쉬운 상황을 만듭니다.

Step 1: 데이터 준비 및 전처리

1. **데이터 불러오기 및 세트 분리:** 교재의 코드처럼 판다스를 이용해 데이터를 불러온 후, 훈련 세트와 테스트 세트로 나눕니다.Python

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('https://bit.ly/perch_csv_data')
perch_full = df.to_numpy()

train_input, test_input, train_target, test_target = train_test_split(
    perch_full, perch_weight, random_state=42)
```

2. **특성 공학 (PolynomialFeatures):** 교재처럼 `PolynomialFeatures`를 사용해 특성의 개수를 의도적으로 늘려 과대적합 가능성이 높은 복잡한 모델을 만듭니다.Python

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=5, include_bias=False)
```

```
poly.fit(train_input)
train_poly = poly.transform(train_input)
test_poly = poly.transform(test_input)
```

3. **데이터 스케일링 (StandardScaler): (매우 중요!)** 규제는 계수의 크기에 직접 페널티를 부과합니다. 만약 특성들의 스케일이 다르다면, 특정 계수만 부당하게 큰 페널티를 받게 됩니다. 따라서 규제를 적용하기 전에는 **반드시 1주차에서 배운 스케일링을 적용**해야 합니다.Python

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(train_poly)
train_scaled = ss.transform(train_poly)
test_scaled = ss.transform(test_poly)
```

Step 2: `alpha` 값 변화에 따른 계수 크기 확인

이제 준비된 데이터에 릿지(Ridge)와 라쏘(Lasso) 모델을 적용해 볼 차례입니다. `alpha` 값을 바꿔가며 각 모델의 계수가 어떻게 변하는지 관찰합니다.

1. **릿지 회귀 (Ridge Regression) 실험:**Python
   - `alpha` 값을 `0.001`, `0.1`, `1`, `10`, `100` 등으로 바꿔가며 모델을 훈련시키고 계수를 출력합니다.

```
from sklearn.linear_model import Ridge

# 테스트할 alpha 값 리스트
alpha_list = [0.001, 0.1, 1, 10, 100]

print("--- 릿지 회귀 계수 변화 ---")
for alpha_val in alpha_list:
    ridge = Ridge(alpha=alpha_val)
    ridge.fit(train_scaled, train_target)

    # alpha 값과 함께 계수 크기의 제곱 평균(L2 norm)을 확인하면 변화를 보기 좋습니다.
    coef_magnitude = np.sum(ridge.coef_**2)
```

```
    print(f"Alpha: {alpha_val}, 계수 제곱 합: {coef_magnitude:.3f}")
    # print(ridge.coef_) # 전체 계수를 직접 확인해봐도 좋습니다.
```

2. **라쏘 회귀 (Lasso Regression) 실험:**Python
    - 릿지와 동일하게 `alpha` 값을 바꿔가며 모델을 훈련시키고 계수를 출력합니다. 특히 **얼마나 많은 계수가 정확히 0이 되는지** 주목해서 보세요.

```
from sklearn.linear_model import Lasso
import numpy as np

print("\n--- 라쏘 회귀 계수 변화 ---")
for alpha_val in alpha_list:
    lasso = Lasso(alpha=alpha_val)
    lasso.fit(train_scaled, train_target)

    # 0이 된 계수의 개수를 세어봅니다.
    zero_coef_count = np.sum(lasso.coef_ == 0)
    print(f"Alpha: {alpha_val}, 0이 된 계수 개수: {zero_coef_count}")
    # print(lasso.coef_)
```

    - `Lasso` 모델에서 `alpha` 값을 크게 할 때 수렴 경고(ConvergenceWarning)가 나타날 수 있습니다. 이는 모델이 최적의 계수를 찾기 위해 반복 계산을 충분히 하지 못했다는 의미이며, `max_iter` 매개변수를 늘려 해결할 수 있습니다. 과제에서는 결과 확인이 목적이므로 일단 무시하고 진행해도 괜찮습니다.

3. 과제 결과 정리 및 공유 (노션, 깃허브 등)

실험 결과를 바탕으로 아래와 같이 정리하여 공유하면 좋습니다.

1. **실험 제목:** 규제 강도(`alpha`) 변화에 따른 선형 회귀 모델 계수 비교
2. **실험 결과 요약 (표 형태로 정리 추천):**

`alpha` 값	릿지(Ridge) - 계수 제곱 합	라쏘(Lasso) - 0이 된 계수 개수
0.001	(계산된 값)	(계산된 값)
0.1	(계산된 값)	(계산된 값)
1	(계산된 값)	(계산된 값)
10	(계산된 값)	(계산된 값)
100	(계산된 값)	(계산된 값)

Sheets로 내보내기

1. **결론 및 분석:**
    - **공통적인 경향:** "릿지와 라쏘 모델 모두에서, 규제 강도를 의미하는 `alpha` 값이 커질수록 모델의 전반적인 계수(coefficient) 크기가 작아지는 경향을 보였다."
    - **이것의 의미:** "이는 `alpha`가 커질수록 모델의 복잡도에 더 강한 페널티가 부과되어, 모델이 과대적합을 피하기 위해 스스로 더 단순한 형태(계수 크기가 작은)가 되려는 것을 의미한다."
    - **릿지와 라쏘의 차이점:**
        - "릿지 회귀는 `alpha`가 커져도 계수 크기를 0에 가깝게 만들 뿐, 완전히 0으로 만들지는 않았다."
        - "반면, 라쏘 회귀는 `alpha`가 커지자 중요도가 낮은 특성의 계수를 **완전히 0으로** 만들어 버렸다. 이는 라쏘 모델이 불필요한 특성을 스스로 제거하는 **특성 선택 (Feature Selection)** 효과가 있음을 보여준다."
    - **최종 결론:** "규제는 과대적합을 제어하는 효과적인 도구이며, `alpha` 값을 통해 강도를 조절할 수 있다. 특히 모델의 특성 전체를 유지하며 과대적합을 완화하고 싶을 때는 **릿지**를, 불필요한 특성을 제거하여 더 간결한 모델을 만들고 싶을 때는 **라쏘**를 사용하는 것이 효과적임을 실험을 통해 확인했다."