기계학습 중간고사 대체과제

프로젝트 보고서

202021497 생명공학과 이주철

- 지도학습을 이용한 회귀 모델 구현 및 분석 -

제출일: 2025-04-20

1. 개발 환경 설정

1) 개발 환경

- 운영체제: Windows 11

- 언어: Python 3.13.2

2) 사용 라이브러리

- pandas: 데이터 불러오기, 정리 (read_csv, DataFrame, 결측치 처리 등)

- Numpy: 수치 계산, 결측값 처리 (np.nan, np.abs 등)

- matplotlib.pyplot: 결과 시각화 (산점도, 히트맵 등)

- seaborn: 상관관계 히트맵 등 고급 시각화

- scikit-learn (sklearn): 머신러닝 전처리, 모델, 평가 전체를 하기 위한 패키지

2. 프로젝트 개요

1) 문제 정의: 냉난방 부하를 건물의 다양한 건축 특성에 따라 예측하는 회귀 문제를 정의합니다.

2) 데이터셋 설명: 이 데이터셋은 건물 면적, 방향, 창의 유무, 크기 등 건축물의 특성을 가진 컬럼과 그 요인들로 인해 냉난방 부하가 어떤 영향을 미쳤는지를 보여주는 수치형 데이터 파일입니다.

- 출처: https://www.kaggle.com/datasets/ujjwalchowdhury/energy-efficiencydata-set

- 데이터셋 크기 (샘플 수 × 변수 수): 768 샘플 * 10 개 변수

- 종속변수(target) 및 주요 독립변수(features):

종속변수: Heating_Load (난방 부하), Cooling_Load (냉방 부하)

독립변수: Relative_Compactness, Surface_Area, Wall_Area, Roof_Area, Overall_Height, Orientation, Glazing_Area, Glazing_Area_Distribution

3. 데이터 전처리 및 탐색적 분석(EDA)

1) 결측치 처리

전체 변수에 대해 결측치 개수를 확인한 결과, 결측지는 존재하지 않았습니다.

따라서 별도의 삭제나 변환 처리는 필요하지 않다고 판단하여 이 과정은 생략하였습니다.

2) 범주형 변수 처리

- 데이터셋에는 object 타입의 범주형 변수가 존재하지 않고 수치형으로 존재하였습니다. 따라서, 범주형 변수 처리는 불필요하다고 판단하였습니다.

3) 스케일링

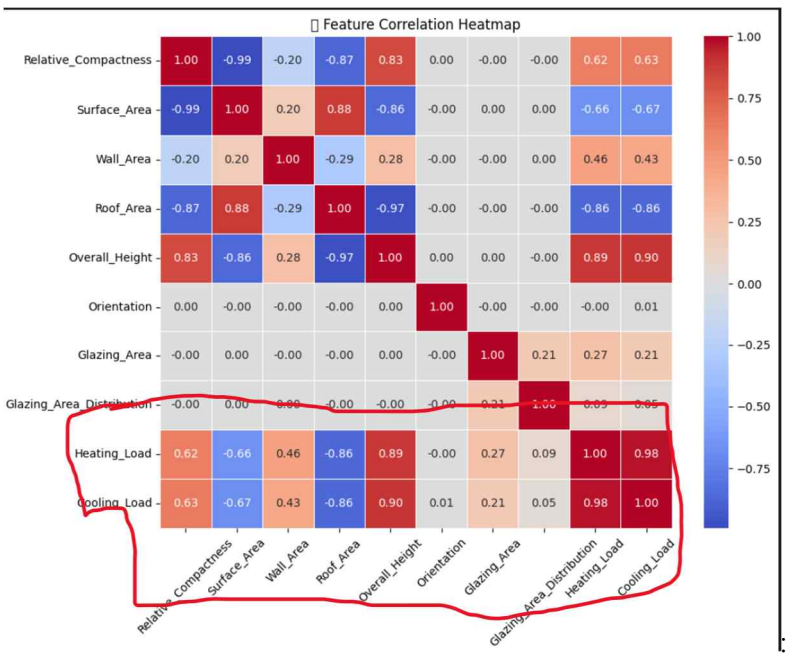
- 사용한 기법: StandardScaler 을 사용하여 입력 피처에 대한 표준화를 적용하였습니다.

- 평균을 0 으로 표준편차를 1 로 표준화를 하여 모델 학습의 수렴 속도 향상과 스케일 차이에 의한 영향 최소화를 위해 사용하였습니다.

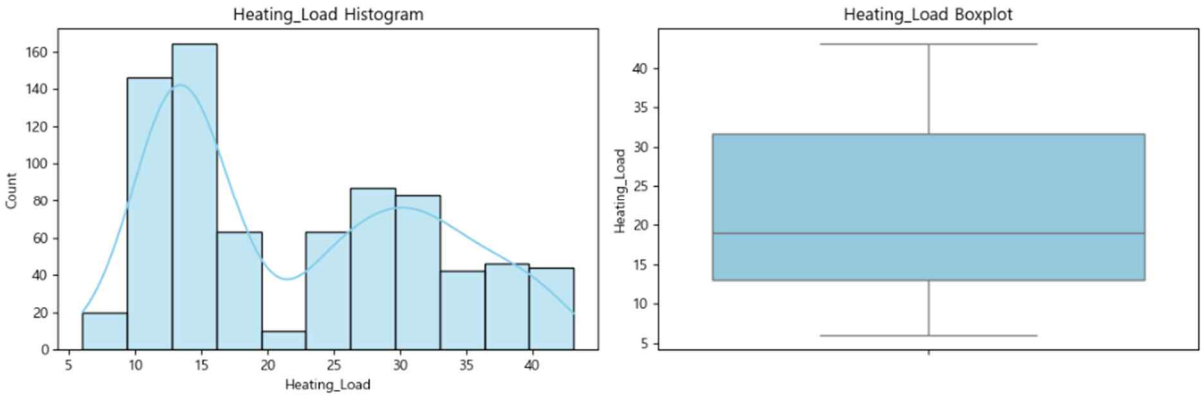
4) EDA 시각화

- Heating_Load, Cooling_Load 와 가장 높은 양의 상관관계를 보이는 변수는 Overall_Height, Relative_Compactness 입니다.

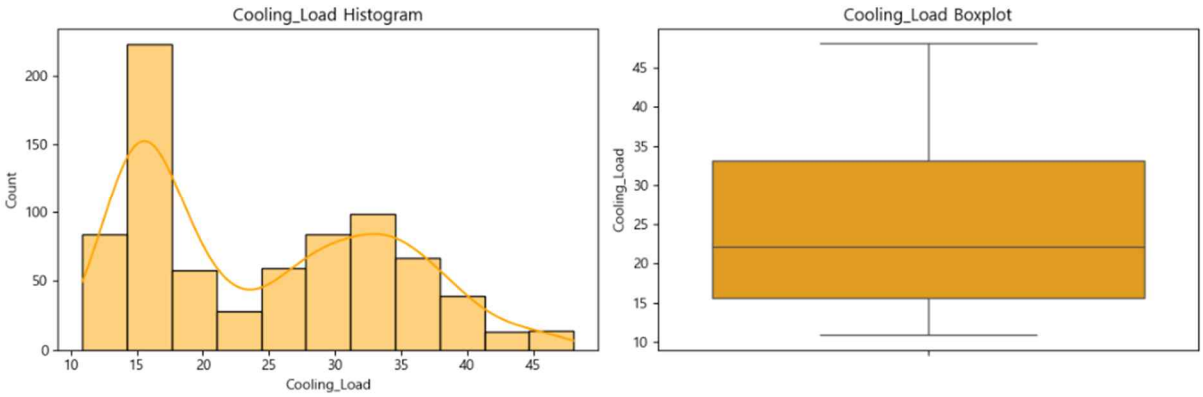
- 종속변수 분포 시각화



: 종속변수와의 상관관계입니다.



: 푸른색이 Heating_Load Histogram 으로 종속변수 분포도를 시각화 해보았습니다. 이는 비대칭적인 분포를 보이고 있으며, 여러 구간에 데이터가 몰려 있다고 판단됩니다.



- 오랜지색인 Cooling_Load Histogram 을 히스토그램으로 시각화 해보았습니다. 이 분포도도 마찬가지로 비대칭적인 분포를 보이고 있으며, 푸른색 히스토그램 역시 왼쪽에 데이터가 몰려있고 오른쪽으로 길게 꼬리가 있는 경우를 보입니다. 이는 Positive skew (양의 왜도)라는 용어를 사용합니다. 혹은 우측 비대칭 (right-skewed)라고도 합니다.

이로써 종속변수의 박스플롯 시각화는 Heating_Load 와 Cooling_Load 모두 박스플롯 기준으로 이상치는 존재하지 않으며, 적절한 모델링의 범위를 가집니다.

4. 모델 구축 및 학습

1) 사용한 알고리즘

- **Linear Regression**: 가장 기본적인 선형 회귀 알고리즘으로, 변수 간의 선형 관계를 가정하고 빠른 학습 속도와 해석 가능성이 장점입니다.

- **Ridge Regression**: L2 정규화를 적용한 선형 회귀로, 과적합을 방지하기 위한 규제 항을 포함하여 일반화 성능을 향상시킵니다.

2) 데이터 분할 방식

- train_test_split 를 사용하여 전체 데이터를 학습용과 테스트용으로 분할하였습니다. 학습 데이터를 이용해 모델을 학습시키고 테스트 데이터로 예측값과 실제값을 비교하여 성능을 평가하였습니다

```
# 2. 과적합을 방지하기 위해 train/test 분할합니다.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

3) 파이프라인 사용 여부

- 파이프라인을 사용함으로써 데이터 전처리와 다중 타겟 회귀 모델을 사용하여 학습과정을 수행하였습니다.

```
# 1. LinearRegression 모델 파이프라인으로 정의합니다.
# - 스케일링 + 선형 회귀 모델로 편의를 고려했습니다.
lr_pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', MultiOutputRegressor(LinearRegression()))
])

# 2. Ridge 모델 파이프라인으로 정의합니다.
# - 정규화를 포함한 선형 모델입니다 .(alph=1.0)
ridge_pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', MultiOutputRegressor(Ridge(alpha=1.0)))
])
```

4) 학습 코드 요약

```
# 1. 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 2. 모델 학습
lr_pipe.fit(X_train, y_train)
ridge_pipe.fit(X_train, y_train)

# 3. 예측 및 평가
y_pred_lr = lr_pipe.predict(X_test)
y_pred_ridge = ridge_pipe.predict(X_test)
```

- 데이터 분할

  -> 전체 데이터를 학습용과 테스트용으로 나눕니다.

- 모델 학습

  -> 학습 데이터를 사용하여 LinearRegression 모델을 학습시킵니다.

  -> 학습 데이터를 사용하여 Ridge 모델을 학습시킵니다.

- 예측 및 평가

  -> 테스트 데이터를 사용하여 LinearRegression 모델로 예측을 수행합니다.

  -> 테스트 데이터를 사용하여 Ridge 모델로 예측을 수행합니다.

5. 성능 평가

1) 사용한 지표

R^2, MAE, RMSE 위 세가지 평가지표를 사용하여 오차를 판단하였습니다.

- R^2, MAE, RMSE 3 가지 지표를 기준으로 데이터 분할에도 안정적으로 예측되고 있는지 객관적으로 보여줍니다.

- 첫번째로는 파이프라인을 구현하고 교차검증을 진행한 결과, 아래에 이미지와 같습니다. 이는 모델 성능의 신뢰성 또는 안정성을 먼저 확인하는데 있어 진행했습니다. 결과는 간접적으로 미리 알 수 없는 데이터를 예측한 결과에 대해 간접적으로 평가된 지표입니다. 교차검증은 Fold 마다 임시로 모델을 만들어서 평가를 진행합니다.

```
[R² 평균]
  LinearRegression R² 평균: 0.8989061150608197
  Ridge R² 평균: 0.8987726425411035

[MAE 평균 (절댓값)]
  LinearRegression MAE 평균: 2.162607182652288
  Ridge MAE 평균: 2.1623923878434113

[RMSE 평균 (절댓값)]
  LinearRegression RMSE 평균: 3.0785156799506113
  Ridge RMSE 평균: 3.080985687300267
```

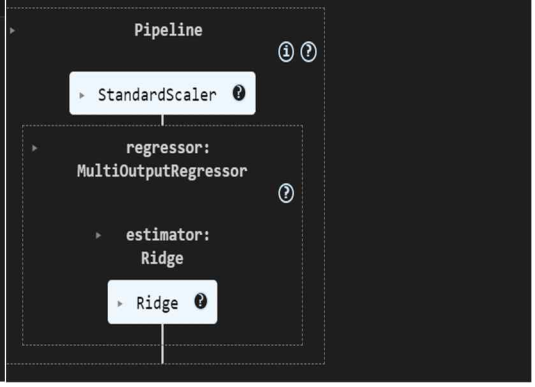
2) 모델 학습을 통한 평가 지표

- .fit()를 실행하여 전처리+모델학습을 진행합니다.

LinearRegression 모델과 Ridge 모델 학습을 수행합니다.

```
# LinearRegression 학습 (기본 선형 회귀 모델)
lr_pipe.fit(X_train_scaled, y_train)
```


```
# Ridge 학습 (L2 정규화를 통해 과적합 방지 효과를 포함한 회귀 모델)
ridge_pipe.fit(X_train_scaled, y_train)
```



- 모델 학습을 수행한 후 지표를 출력합니다.

```
print(f'LinearRegression 검증 데이터 "R²" 점수: {r2_score(y_test, y_pred_lr):.4f}')
print(f'Ridge 검증 데이터 R² 점수: {r2_score(y_test, y_pred_ridge):.4f}')
```

Python

```
LinearRegression 검증 데이터 "R²" 점수: 0.9021
Ridge 검증 데이터 R² 점수: 0.9016
```

```
print(f'LinearRegression 검증 데이터 "RMSE" 점수: {mean_squared_error(y_test, y_pred_lr):.4f}')
print(f'Ridge 검증 데이터 "RMSE" 점수: {mean_squared_error(y_test, y_pred_ridge):.4f}')
```

Python

```
LinearRegression 검증 데이터 "RMSE" 점수: 9.5860
Ridge 검증 데이터 "RMSE" 점수: 9.6367
```

```
print(f'LinearRegression 검증 데이터 "MAE" 점수: {mean_absolute_error(y_test, y_pred_lr):.4f}')
print(f'Ridge 검증 데이터 "MAE" 점수: {mean_absolute_error(y_test, y_pred_ridge):.4f}')
```

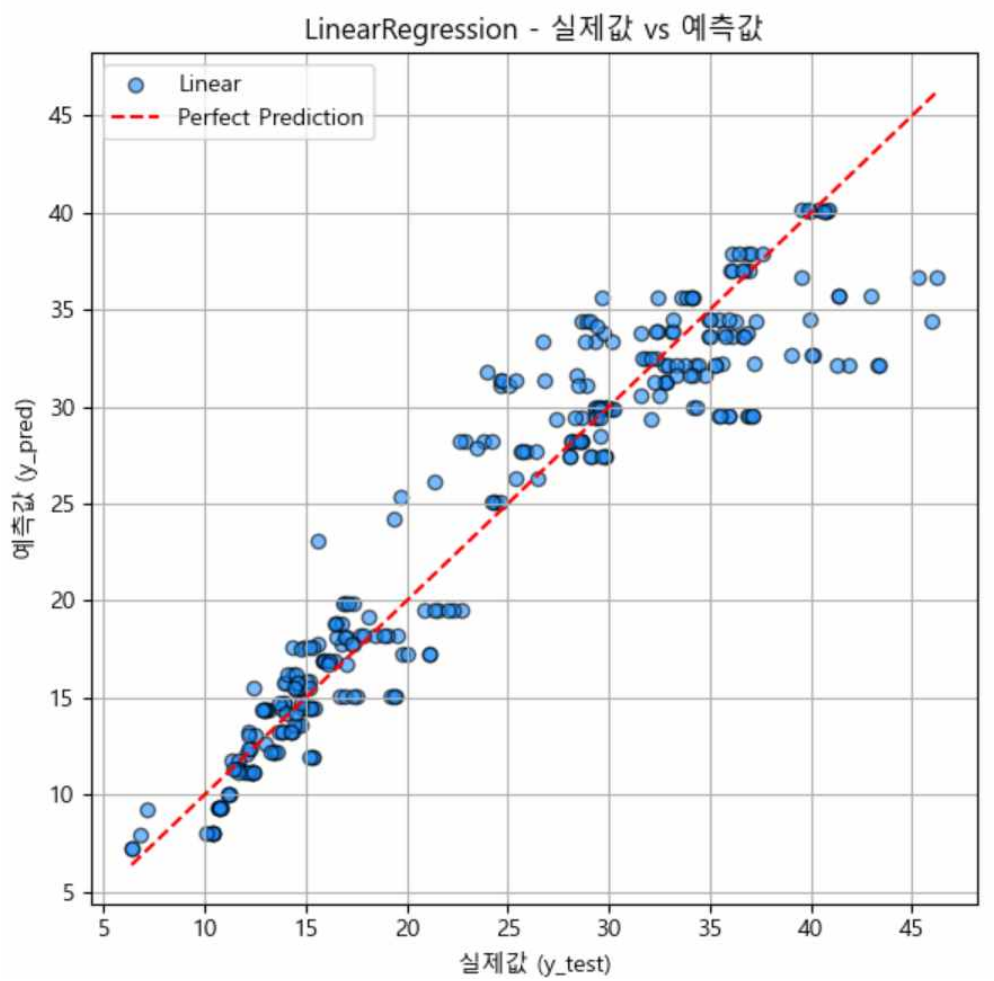
Python

```
LinearRegression 검증 데이터 "MAE" 점수: 2.1824
Ridge 검증 데이터 "MAE" 점수: 2.1834
```

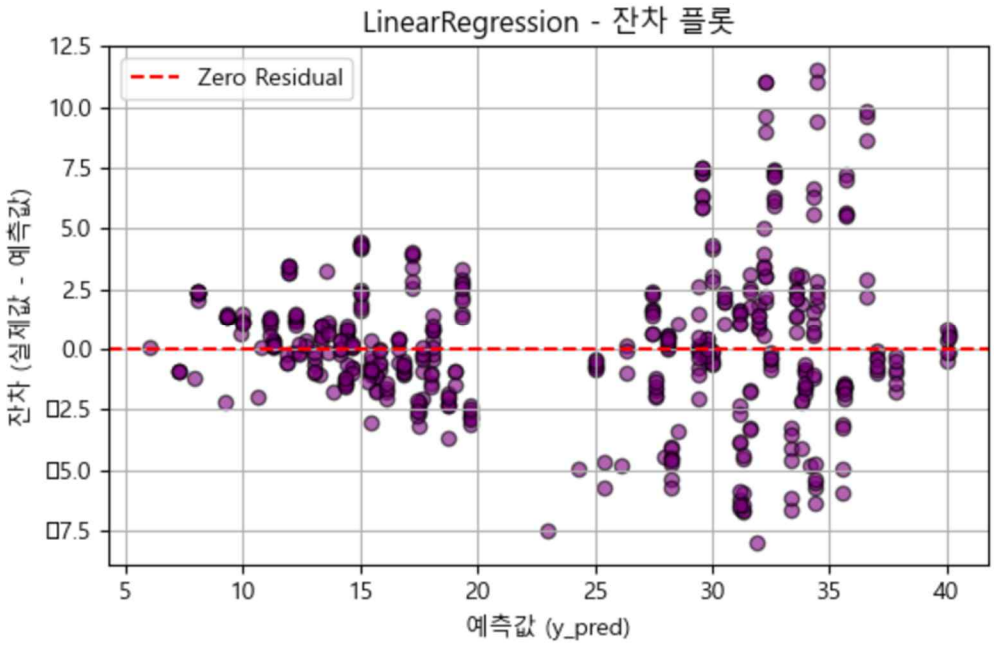
LinearRegression 과 Ridge 모델을 최종 회귀 모델로 선정하여 학습을 진행합니다.
.fit() 메서드를 수행하여 테스트 데이터를 기반으로 학습시킨 후, 테스트 데이터를 예측을 수행하였습니다.
평가 후 예측 결과는 평가 지표를 기준으로 평가하였고, 위와 같은 성능을 보여줍니다.

3) 예측값 vs 실제값 시각화

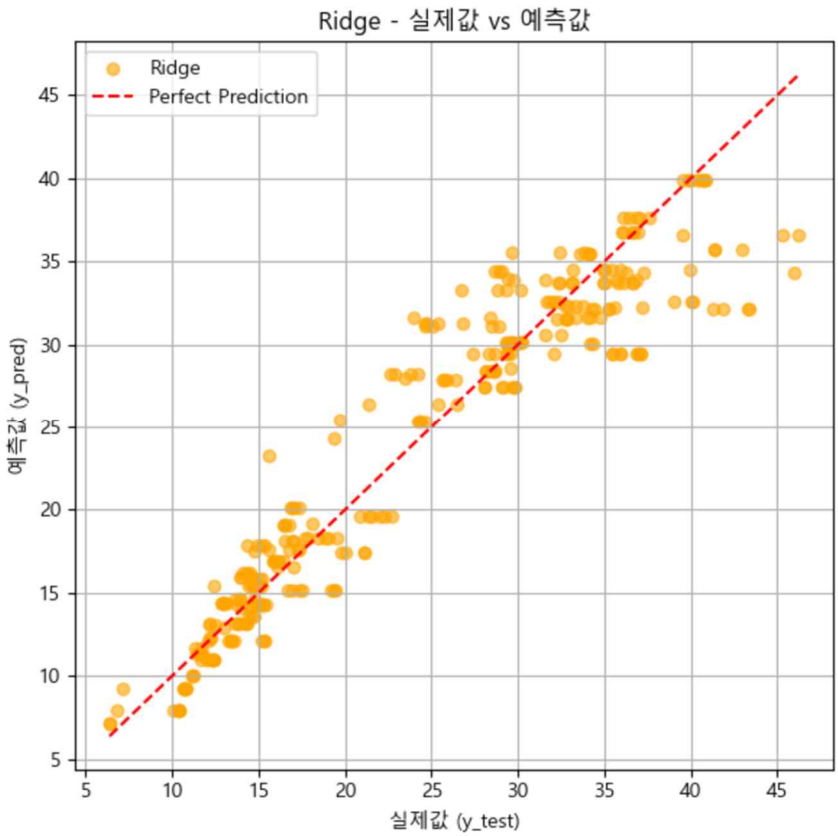
시각화를 통해 예측이 어느정도 인지 평가합니다.



- 위 이미지에서 LinearRegression 모델을 통해 실제값과 예측값을 산점도로 시각화 합니다. 빨간 점선이 완변한 예측 정도를 말하는 것이며, 선 위에 파란 점들이 가까울수록 예측이 정확한 것입니다. 선보다 위에 있으면 과대 예측이며, 보다 아래에 있을 경우 과소 예측이라고 표현합니다. 점들이 선에서 멀리 떨어진 경우는 모델의 에측 정도가 떨어진다라고 판단할 수 있습니다.

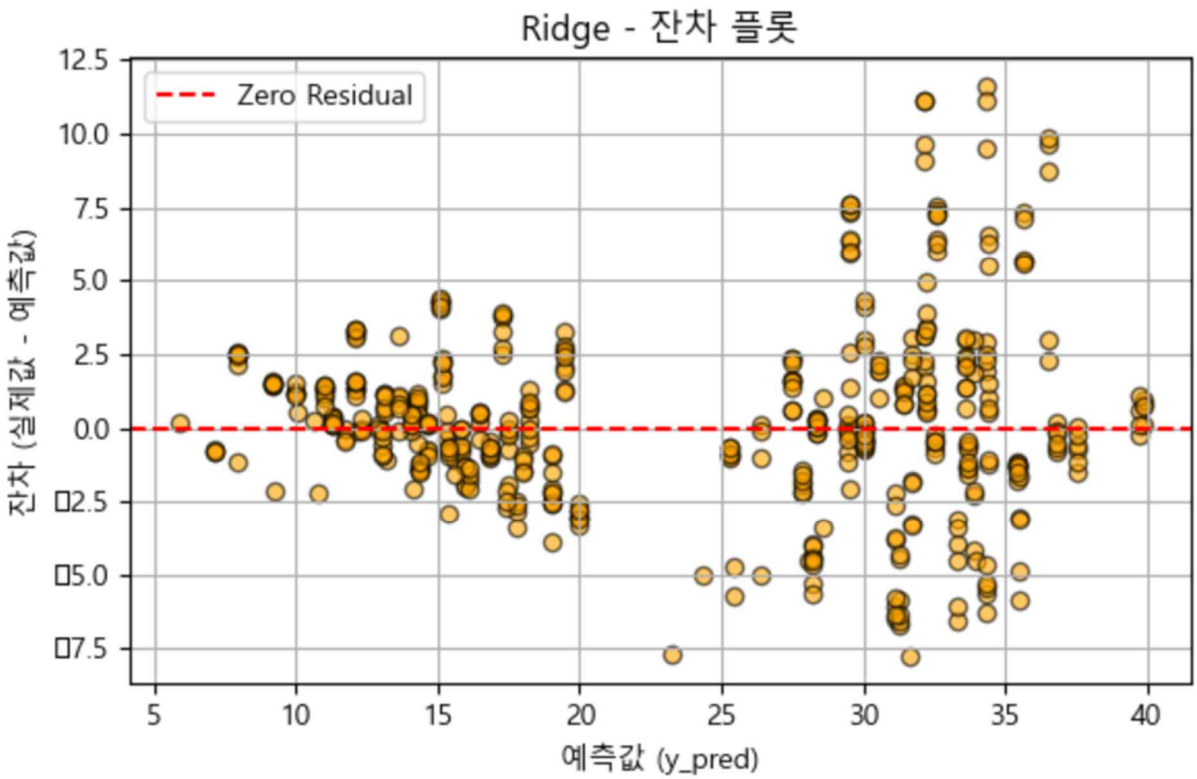


- 위 이미지에서 LinearRegression 모델을 활용해서 잔차 플롯으로 시각화를 진행합니다. 이는 빨간 점선을 기준으로 오차 범위를 점선으로 확인할 수 있습니다.



- 위 이미지에서 Ridge 모델을 통해 실제값과 예측값을 산점도로 시각화 합니다. 주황색 점선이 완변한 예측 정도를 말하는 것이며, 선 위에 주황 점들이 가까울수록 예측이

정확한 것입니다. 선보다 위에 있으면 과대 예측이며, 보다 아래에 있을 경우 과소 예측이라고 표현합니다. 점들이 선에서 멀리 떨어진 경우는 모델의 에측 정도가 떨어진다라고 판단할 수 있습니다. LinearRegression 모델과 거의 유사한 분포도를 띈다고 볼 수 있습니다.



- 위 이미지에서 Ridge 모델을 활용해서 잔차 플롯으로 시각화를 진행합니다. 빨간 점선으로 기준으로 점들의 분포를 보고 오차가 어느 정도로 분포 되어있는지 확인할 수 있습니다.

4) 해석:

- 전반적으로 데이터가 완벽한 직선 위에 위치하진 않지만, 큰 편차 없이 분포하고 있어 과소/과대 예측의 편향도 크지 않음을 확인할 수 있습니다. 두 모델 모두 선형선과 높은 예측 정밀도를 보인다고 해석할 수 있습니다.
- 과소/과대 예측은 적은 편으로 판단되고, 선에서 크게 벗어난 점은 소수라고 생각하여 예측이 잘되었다고 봅니다. 다만, 40 이상 영역에서는 점들이 다소 퍼져있는 것을 확인할 수 있습니다. 이 구간에서는 예측이 약간 덜 정확하다고 생각합니다.

6. 하이퍼파라미터 튜닝

1) 튜닝 방법

- 모델의 성능을 높이기 위해 GridSearchCV, RandomizedSearchCV 두 가지 방법을 사용하여 하이퍼파라미터를 탐색하였습니다.

- cv=5 로 교차검증을 적용시켜 성능이 저하되지 않도록 최적값을 안정적으로 나오게 진행하였습니다. 평가 지표 3 가지를 통해 최적의 하이퍼파라미터를 도출하였습니다.

2)튜닝한 하이퍼파라미터

- 사용한 모델은 Ridge 입니다. Linearregression 은 데이터만으로 최적의 가중치를 계산하고 튜닝 없이 예측값을 구하는 모델이기에 따로 지정하지 않고 진행하였습니다.

- 주요 하이퍼파라미터는 alpha 로 설정하였습니다. L2 정규화 강도를 조절하는 파라미터로 과적합을 방지하는데 도움이 되어 튜닝을 진행하였습니다.

```
from sklearn.model_selection import GridSearchCV

# 하이퍼파라미터 범위 설정
# alpha는 Ridge의 L2 정규화 강도를 조절하는 파라미터입니다. 값을 정규화 시켜 과적합을 방지합니다.
# 로그 스케일로 실험하는 것이 효과적이라고 생각하여 범위를 10배 단위로 설정하였습니다.
param_grid = {
    'regressor__estimator__alpha': [0.01, 0.1, 1.0, 10.0, 100.0]
}

# GridSearchCV 수행 cv=5 교차검증으로 apha 최적값을 탐색합니ㅏ다.
grid = GridSearchCV(ridge_pipe, param_grid, cv=5, scoring='r2')
grid.fit(X_train, y_train)
```

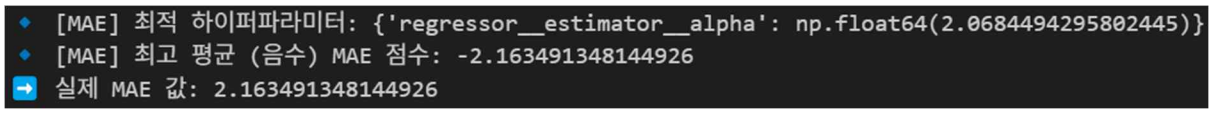
: GridSearchCV 를 사용하여 최적의 하이퍼파라미터를 탐색합니다.

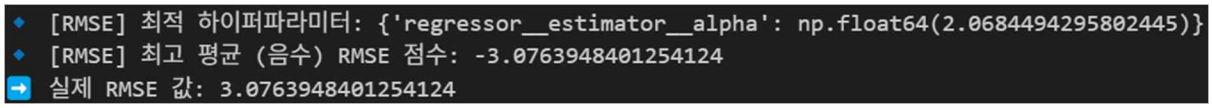
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform

# 탐색할 alpha 값의 범위를 정의해줍니다.
# 0.01 ~ 100 범위 안에서 무작위로 샘플링을 합니다.
param_dist = {
    'regressor__estimator__alpha': uniform(loc=0.01, scale=100)
}
```

: RandomizedSearchCV 를 사용하여 최적의 하이퍼파라미터를 탐색합니다.

3)튜닝 결과 분석

- GridSearchCV 결과는 아래 이미지와 같이 평가 지표 R^2, MAE, RMSE 로 분석한 것입니다.





- RandomizedSearchCV 결과는 아래 이미지와 같이 평가 지표 R^2, MAE, RMSE 로 분석한 것입니다.

7. 결론 및 고찰

1) 최종 모델 성능 종합 평가

- 본 프로젝트에서는 LinearRegression 과 Ridge 회귀 모델을 사용하여 건물의 특성으로부터 냉난방 부하를 예측하는 프로그램을 만들어보았습니다. 교차검증 및 하이퍼파라미터를 설정하여 나온 평가 결과는 **R^2 평균 점수**: 약 **0.89 ~ 0.90, MAE (평균 절대 오차): 약 2.1, RMSE (제곱 평균 오차의 제곱근): 약 3.0 으로** 예측값과 실제값을 비교하여 시각화를 통해 모델이 전반적으로 안정적인 성능을 보였다고 판단하였습니다.

2) 데이터 또는 모델의 한계

- LinearRegression, Ridge 모두 선형성을 가정하기 때문에 **비선형적인 영향 요인**을 충분히 반영하지 못할 수도 있다는 한계를 만날 수 있습니다. 창면적이 커질수록 냉방 부하가 비선형적으로 증가할 수도 있고 혹은 층고가 높을수록 냉난방 부하가 비선형적으로 증가할 수 있는 경우가 있을 수도 있습니다. 사실 현실 세계의 변수들은 선형적보다는 비선형적인 관계가 더 많다고 생각합니다. 따라서 위 컬럼들이 단순한 선형으로 비례하지 않을 수 있는 점이 한계라고 봅니다.

3) 실생활 응용 가능성 또는 확장 방향

- 건물 특징을 가지고 냉난방 부하를 예측하는 프로젝트는 건축 설계 초기 단계에서 냉난방 부하를 미리 예측하고 설계를 진행할 수 있다고 생각합니다. 어느 부분을 조정하여 에너지 효율을 극대화시킬 수 있는지도 판단할 수 있을 것입니다.

4) 다음 단계에서 고려할 점 (특성 선택, 앙상블, 이상치 제거 등)

- 시각화를 통해 상관관계를 파악하고 그 중에서 상관관계가 낮거나 오히려 성능을 저해하는 변수를 제거해야 성능 향상이 가능하다고 생각합니다. 하지만 낮다고 삭제했을 때 오히려 결과가 좋지 않게 나와 그런 점도 비교하면서 고려해야 합니다.

- 비선형적 모델을 사용하여 예측 정확도에 대한 성능 향상 가능성이 존재하기 때문에 이를 도입하는 방법도 고려할 점이라고 생각합니다.

- 데이터의 오차 범위를 줄이기 위해 시각화를 통해 상관관계와 오차의 분포를 파악하고 이상치를 제거하는 것을 고려해야 합니다.

8. 참고자료

- https://white-joy.tistory.com/10 (회귀 모델 성능 평가 지표 설명)

- https://www.kaggle.com/datasets/ujjwalchowdhury/energy-efficiencydata-set (데이터셋 출처 URL)

- 혼자 공부하는 머신러닝+딥러닝 (도서관 교재 참조)

- Data camp Supervised Learning with scikit-learn (강의 자료 참고)

- https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/regression-23254832-2f6e-45bb-a1bc-11dbe2461668?ex=12

- ChatGPT (코드정리 및 문단정리를 하기 위해 사용)