6. 냉난방 부하예측 머신러닝모델 개발

박경규

냉난방 부하 예측모델

건물 에너지 소비는 전체 에너지 사용의 약 40%를 차지합니다.
설계 단계에서 건물의 냉난방 부하를 예측하는 것은 에너지 효율화를 위해 매우 중요합니다.
건물 에너지 성능에 대한 데이터셋을 기반으로 건물의 냉난방 부하를 예측하는 다양한 모델을 개발합니다.

■ Input

- Relative Compactness
- Surface Area - m^2
- Wall Area - m^2
- Roof Area - m^2
- Overall Height - m
- Orientation - 2:North, 3:East, 4:South, 5:West
- Glazing Area - 0%, 10%, 25%, 40% (of floor area)
- Glazing Area Distribution (Variance)
  - 1:Uniform, 2:North, 3:East, 4:South, 5:West
- Heating Load - kWh
- Cooling Load - kWh

■ Output

- 난방부하
- 냉방부하

냉난방 부하 예측모델

energy_efficiency_modeling.ipynb

[1]

```python
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]

```python
data = pd.read_csv('building_energy_efficiency.csv')
```

[3]

```python
data.head()
```

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load	Cooling Load
0	0.7638	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.9800	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.9800	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.9800	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.9000	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28

냉난방 부하 예측모델

```
[4]  data.shape
```

```
(768, 10)
```

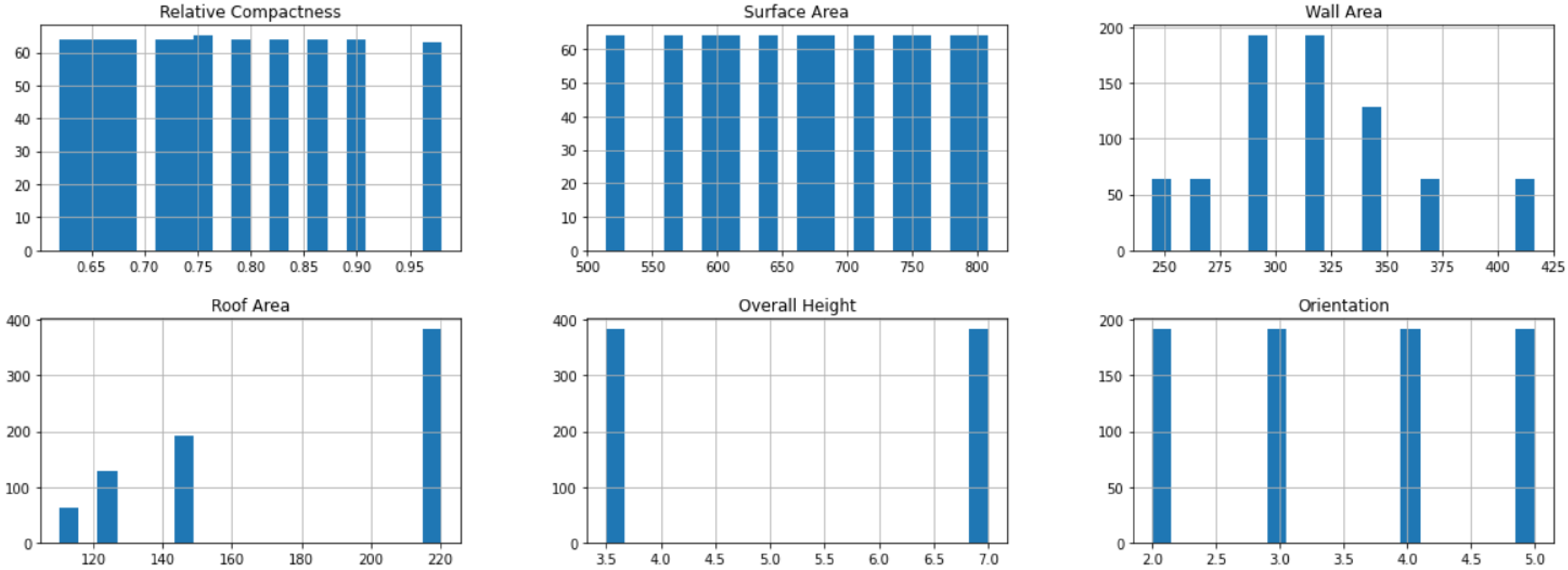
```
[5]  data.isnull().sum()
```

```
Relative Compactness         0
Surface Area                 0
Wall Area                    0
Roof Area                    0
Overall Height               0
Orientation                  0
Glazing Area                 0
Glazing Area Distribution    0
Heating Load                 0
Cooling Load                 0
dtype: int64
```

냉난방 부하 예측모델

각 데이터의 분포 체크

```
[6]  data.hist(bins=20, figsize=(20,15))
     plt.show()
```

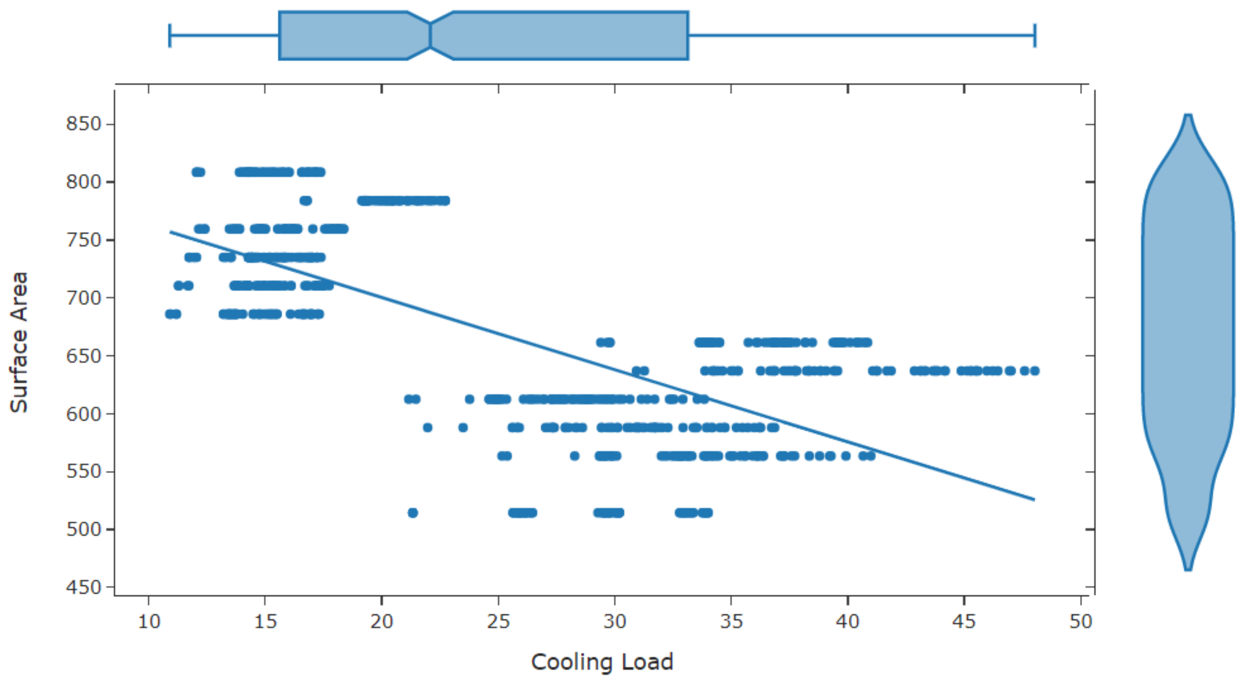


냉난방 부하 예측모델

상관관계 분석

```
[7]  import plotly.express as px

     yprop = 'Surface Area'
     xprop = 'Cooling Load'
     h= None
     px.scatter(data, x=xprop, y=yprop, color=h, marginal_y="violin", marginal_x="box", trendline="ols",
```



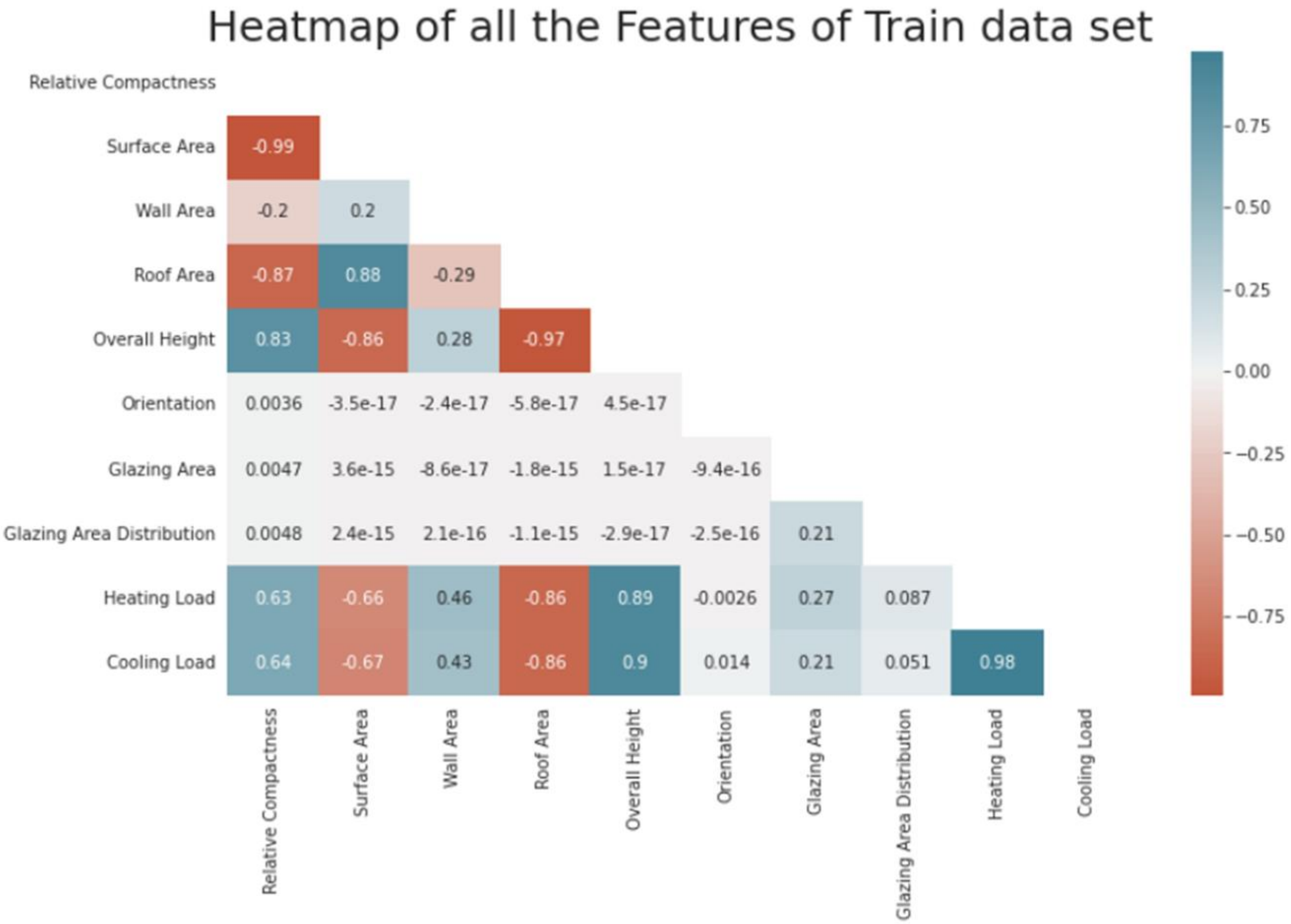
냉난방 부하 예측모델

명확한 상관관계를 찾기위해 상관관계 행렬 확인

```
[9]  import matplotlib.pyplot as plt
     import matplotlib.style as style
     import seaborn as sns
     style.use('ggplot')
     sns.set_style('whitegrid')

     plt.subplots(figsize = (12,7))
     ## Plotting heatmap. # Generate a mask for the upper triangle (taken from seabor
     mask = np.zeros_like(data.corr(), dtype=np.bool)
     mask[np.triu_indices_from(mask)] = True
     sns.heatmap(data.corr(), cmap=sns.diverging_palette(20, 220, n=200), annot=True,
     plt.title("Heatmap of all the Features of Train data set", fontsize = 25);
```

냉난방 부하 예측모델


Heatmap of all the Features of Train data set
Relative Compactness
Surface Area
Wall Area
Roof Area
Overall Height
Orientation
Glazing Area
Glazing Area Distribution
Heating Load
Cooling Load
-0.99
-0.2
0.2
-0.87
0.88
-0.29
0.83
-0.86
0.28
-0.97
0.0036
-3.5e-17
-2.4e-17
-5.8e-17
4.5e-17
0.0047
3.6e-15
-8.6e-17
-1.8e-15
1.5e-17
-9.4e-16
0.0048
2.4e-15
2.1e-16
-1.1e-15
-2.9e-17
-2.5e-16
0.21
0.63
-0.66
0.46
-0.86
0.89
-0.0026
0.27
0.087
0.64
-0.67
0.43
-0.86
0.9
0.014
0.21
0.051
0.98
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

냉난방 부하 예측모델

```
[11] from scipy.stats import randint as sp_randint
     from catboost import CatBoostRegressor
     from sklearn.model_selection import GridSearchCV
     from keras.layers import Dense
     from keras.models import Sequential
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.neighbors import KNeighborsRegressor
     from sklearn.neural_network import MLPRegressor
     from sklearn.ensemble import GradientBoostingRegressor,AdaBoostRegressor
     from sklearn.ensemble import BaggingRegressor, RandomForestRegressor
     from sklearn.model_selection import GridSearchCV
     from sklearn.model_selection import train_test_split
     from sklearn.multioutput import MultiOutputRegressor
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.svm import SVC
     from sklearn.svm import SVR
     from sklearn.metrics import accuracy_score, f1_score
     from sklearn.metrics import r2_score
     from sklearn.metrics import roc_auc_score
```

냉난방 부하 예측모델

```
[12] X = data[['Relative Compactness', 'Surface Area', 'Wall Area',
               'Roof Area', 'Overall Height', 'Orientation',
               'Glazing Area', 'Glazing Area Distribution']]
     Y = data[['Heating Load', 'Cooling Load']]
     Y1= data[['Heating Load']]
     Y2= data[['Cooling Load']]
```

냉난방 부하 예측모델

- 데이터 세트를 훈련 및 테스트 세트로 분할.
- 특징 스케일링 또는 데이터 정규화는 데이터의 독립 변수 또는 특징의 범위를 정규화하는 데 사용되는 방법입니다.
- 따라서 독립 변수에서 값이 많이 다를 때 모든 값이 비교 가능한 범위에 유지되도록 특성 스케일링을 사용합니다.

```
[13] X_train, X_test, y1_train, y1_test, y2_train, y2_test = train_test_split(
         X, Y1, Y2, test_size=0.33, random_state = 20)

     scaler = MinMaxScaler(feature_range= (0,1))
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)
```

냉난방 부하 예측모델

모델링

각 모델의 결과를 저장할 DataFrame을 만듭니다.

```
[14]  accuracy = pd.DataFrame(index=None, columns=['model',
                                                  'train_Heating','test_Heating',
                                                  'train_Cooling','test_Cooling'])
```

```
[15]  regressors = [['SVR',SVR()],
                    ['DecisionTreeRegressor',DecisionTreeRegressor()],
                    ['KNeighborsRegressor', KNeighborsRegressor()],
                    ['RandomForestRegressor', RandomForestRegressor()],
                    ['MLPRegressor',MLPRegressor()],
                    ['AdaBoostRegressor',AdaBoostRegressor()],
                    ['GradientBoostingRegressor',GradientBoostingRegressor()]]
```

냉난방 부하 예측모델

```
[16] for r in regressors:
         name = r[0]
         model = r[1]

         model.fit(X_train, y1_train)
         actr1 = r2_score(y1_train, model.predict(X_train))
         acte1 = r2_score(y1_test, model.predict(X_test))

         model.fit(X_train, y2_train)
         actr2 = r2_score(y2_train, model.predict(X_train))
         acte2 = r2_score(y2_test, model.predict(X_test))

         accuracy = accuracy.append(pd.Series({'model':name, 'train_Heating':actr1,
                                     'test_Heating':acte1,'train_Cooling':actr2,
                                     'test_Cooling':acte2}),ignore_index=True )

     accuracy.sort_values(by='test_Cooling')
```

냉난방 부하 예측모델

	model	train_Heating	test_Heating	train_Cooling	test_Cooling
4	MLPRegressor	0.866251	0.874380	0.808819	0.831392
0	SVR	0.930662	0.910593	0.892578	0.887385
2	KNeighborsRegressor	0.945989	0.904490	0.926869	0.888896
1	DecisionTreeRegressor	1.000000	0.995243	1.000000	0.948064
5	AdaBoostRegressor	0.970024	0.964384	0.946378	0.948302
3	RandomForestRegressor	0.999497	0.997359	0.995431	0.964139
6	GradientBoostingRegressor	0.998173	0.997641	0.979423	0.976044

냉난방 부하 예측모델

모델 파라미터 튜닝

- Boosting machine learning 알고리즘은 단순한 알고리즘보다 더 나은 정확도를 제공하기 때문에 많이 사용됩니다.
- 알고리즘의 성능은 하이퍼파라미터에 따라 다릅니다.
- 최적의 parameter는 더 높은 정확도를 달성하는 데 도움이 될 수 있습니다.
- 수동으로 hyper parameter를 찾는 것은 지루하고 계산 비용이 많이 듭니다.
- 하이퍼파라미터 튜닝의 자동화가 중요합니다.
- RandomSearch, GridSearchCV, Bayesian optimization은 하이퍼파라미터를 최적화하는 데 사용됩니다.

냉난방 부하 예측모델

```
[17] param_grid = [{"learning_rate": [0.01, 0.02, 0.1],
                    "n_estimators":[150, 200, 250], "max_depth": [4, 5, 6],
                    "min_samples_split":[1, 2, 3], "min_samples_leaf":[2, 3],
                    "subsample":[1.0, 2.0]}]

     gb = GradientBoostingRegressor()
     gs_gb = GridSearchCV(gb, param_grid, cv=10,
                          scoring='neg_mean_squared_error')
     gs_gb.fit(X_train, y2_train)

     print("R-Squared : {}".format(gs_gb.best_score_))
     print(f"Best Hyperparameters : ₩n{gs_gb.best_params_}")
```

```
R-Squared : -1.0361285669594182
Best Hyperparameters :
{'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 3,
```

냉난방 부하 예측모델

A. Decision Tree Regressor parameters turning

```
[18]  dt = DecisionTreeRegressor()
      param_grid = {"criterion": ["squared_error", "absolute_error"],"min_samples_split": [15, 1(
                    "max_depth": [5, 6, 7],"min_samples_leaf": [4, 5, 6],
                    "max_leaf_nodes": [29, 30, 31, 32],}

      gs_dt = GridSearchCV(dt, param_grid, cv=5)

      gs_dt.fit(X_train, y2_train)
      print(f"R-Squared : {gs_dt.best_score_}")
      print(f"Best Hyperparameters : ₩n{gs_dt.best_params_}")
```

```
R-Squared : 0.959889622709387
Best Hyperparameters :
{'criterion': 'squared_error', 'max_depth': 6, 'max_leaf_nodes': 30, 'min_samples_leaf': 5
```

냉난방 부하 예측모델

```
[19] dt = DecisionTreeRegressor(criterion='squared_error', max_depth=6,
                                max_leaf_nodes=32, min_samples_leaf= 5,
                                min_samples_split=17)


     dt.fit(X_train,y1_train)
     print(f"R-Squared on train dataset={dt.score(X_test,y1_test)}")

     dt.fit(X_train,y2_train)
     print(f"R-Squaredon test dataset={dt.score(X_test,y2_test)}")
```

```
R-Squared on train dataset=0.9953559923756888
R-Squaredon test dataset=0.9592409931240966
```

냉난방 부하 예측모델

B. Tune Random Forests Parameters

```
[20] from sklearn.model_selection import GridSearchCV
     param_grid = [{'n_estimators': [350, 400, 450], 'max_features': [1, 2],
                    'max_depth': [85, 90, 95]}]

     rf = RandomForestRegressor(n_jobs=-1)
     gs_rf = GridSearchCV(rf, param_grid, cv=10,
                          scoring='neg_mean_squared_error')
     gs_rf.fit(X_train, y2_train)

     print(f"R-Squared::{gs_rf.best_score_}")
     print(f"Best Hyperparameters::\n{gs_rf.best_params_}")
```

```
R-Squared::-2.704375462246052
Best Hyperparameters::
{'max_depth': 95, 'max_features': 1, 'n_estimators': 450}
```

냉난방 부하 예측모델

```
[21] rf = RandomForestRegressor(n_estimators=250, max_features=1,
                                max_depth=90, bootstrap=True)

     rf.fit(X_train,y1_train)
     print("R-Squared on train dataset={}".format(rf.score(X_test,y1_test)))

     rf.fit(X_train,y2_train)
     print(f"R-Squaredon test dataset={rf.score(X_test,y2_test)}")
```

```
R-Squared on train dataset=0.9910854073670696
R-Squaredon test dataset=0.9743420800363577
```

냉난방 부하 예측모델

C. Gradient Boosting Regression - Hyperparameter Tuning

```
[22] gbr = GradientBoostingRegressor(learning_rate=0.1,n_estimators=250,
                                     max_depth=5, min_samples_split=3,
                                     min_samples_leaf=2, subsample=1.0)

     gbr.fit(X_train,y1_train)
     print(f"R-Squared on train dataset={gbr.score(X_test,y1_test)}")

     gbr.fit(X_train,y2_train)
     print(f"R-Squaredon test dataset={gbr.score(X_test,y2_test)}")
```

```
R-Squared on train dataset=0.9986719997364437
R-Squaredon test dataset=0.9915692478361576
```

냉난방 부하 예측모델

D. CatBoostRegressor

```
[23] import warnings
     warnings.filterwarnings("ignore")
     from sklearn import datasets
     from sklearn.model_selection import train_test_split
     from sklearn.model_selection import GridSearchCV
     from catboost import CatBoostRegressor

     cb = CatBoostRegressor()
     parameters = {'depth':[8, 10],'iterations':[10000],'learning_rate':[0.02,0.03],
                   'border_count':[5],'random_state': [42, 45]}

     gs_cb = GridSearchCV(estimator=cb, param_grid = parameters, cv = 2, n_jobs=-1)
     gs_cb.fit(X_train, y2_train)
     print("Results from Grid Search" )
     print(f"₩n The best estimator across ALL searched params:₩n{gs_cb.best_estimator_}")
     print(f"₩n The best score across ALL searched params:₩n{gs_cb.best_score_}")
```

```
[24] print(f"₩n The best parameters across ALL searched params:₩n{gs_cb.best_params_}")
```

```
 The best parameters across ALL searched params:
{'border_count': 5, 'depth': 10, 'iterations': 10000, 'learning_rate': 0.02, 'random_state': 4
```

냉난방 부하 예측모델

모델 성능 Summary

```
[25]  accuracy = pd.DataFrame(index=None, columns=['model',
                                                   'train_Heating','test_Heating',
                                                   'train_Cooling','test_Cooling'])
```

```
[26]  regressors = [
          ["DecisionTreeRegressor",
              DecisionTreeRegressor(
                  criterion='squared_error', max_depth=6, max_leaf_nodes=31,
                  min_samples_leaf=5, min_samples_split=17
              ),
          ],
          ["RandomForestRegressor",
              RandomForestRegressor(
                  max_depth=85, max_features=1, n_estimators=350
              ),
          ],
          ["GradientBoostingRegressor",
              GradientBoostingRegressor(
                  learning_rate=0.1, n_estimators=250, max_depth=5,
                  min_samples_split=2, min_samples_leaf=3, subsample=1.0,

              ),
          ],
          ["CatBoostRegressor",
              CatBoostRegressor(
                  border_count=5, depth=10, iterations=10000,
                  learning_rate=0.02, random_state=42
              ),
          ],
      ]
```

냉난방 부하 예측모델

- 난방 및 냉방 부하를 예측하기 위해 다양한 모델을 사용하여 건물 에너지 성능을 예측 했습니다.
- 모델에서 다양한 parameter를 조정하여 매우 좋은 예측 결과를 얻었습니다.
- 난방 및 냉방 부하 모두에서 >99.5%, 실험 데이터 세트와 비교

```
[28] best_model = CatBoostRegressor(border_count=5, depth=10, iterations=10000,
                                    learning_rate=0.02, random_state=42)
     best_model.fit(X_train,y1_train)
     actr1 = r2_score(y1_train, best_model.predict(X_train))
     acte1 = r2_score(y1_test, best_model.predict(X_test))
     y1_pred = best_model.predict(X_test)

     best_model.fit(X_train,y2_train)
     actr2 = r2_score(y2_train, best_model.predict(X_train))
     acte2 = r2_score(y2_test, best_model.predict(X_test))
     y2_pred = best_model.predict(X_test)
```

냉난방 부하 예측모델

	model	train_Heating	test_Heating	train_Cooling	test_Cooling
0	DecisionTreeRegressor	0.994874	0.995248	0.967755	0.959161
1	RandomForestRegressor	0.998717	0.991695	0.996149	0.972668
2	GradientBoostingRegressor	0.999735	0.998552	0.998988	0.991906
3	CatBoostRegressor	1.000000	0.998466	1.000000	0.993174

냉난방 부하 예측모델

```
[29] print("CatBoostRegressor: R-Squared on train dataset={}".format(actr1))
     print("CatBoostRegressor: R-Squared on test dataset={}".format(acte1))
     print("CatBoostRegressor: R-Squared on train dataset={}".format(actr2))
     print("CatBoostRegressor: R-Squared on test dataset={}".format(acte2))
```

```
CatBoostRegressor: R-Squared on train dataset=0.9999999999782324
CatBoostRegressor: R-Squared on test dataset=0.9984655722017103
CatBoostRegressor: R-Squared on train dataset=0.9999999999364746
CatBoostRegressor: R-Squared on test dataset=0.9931744902917424
```

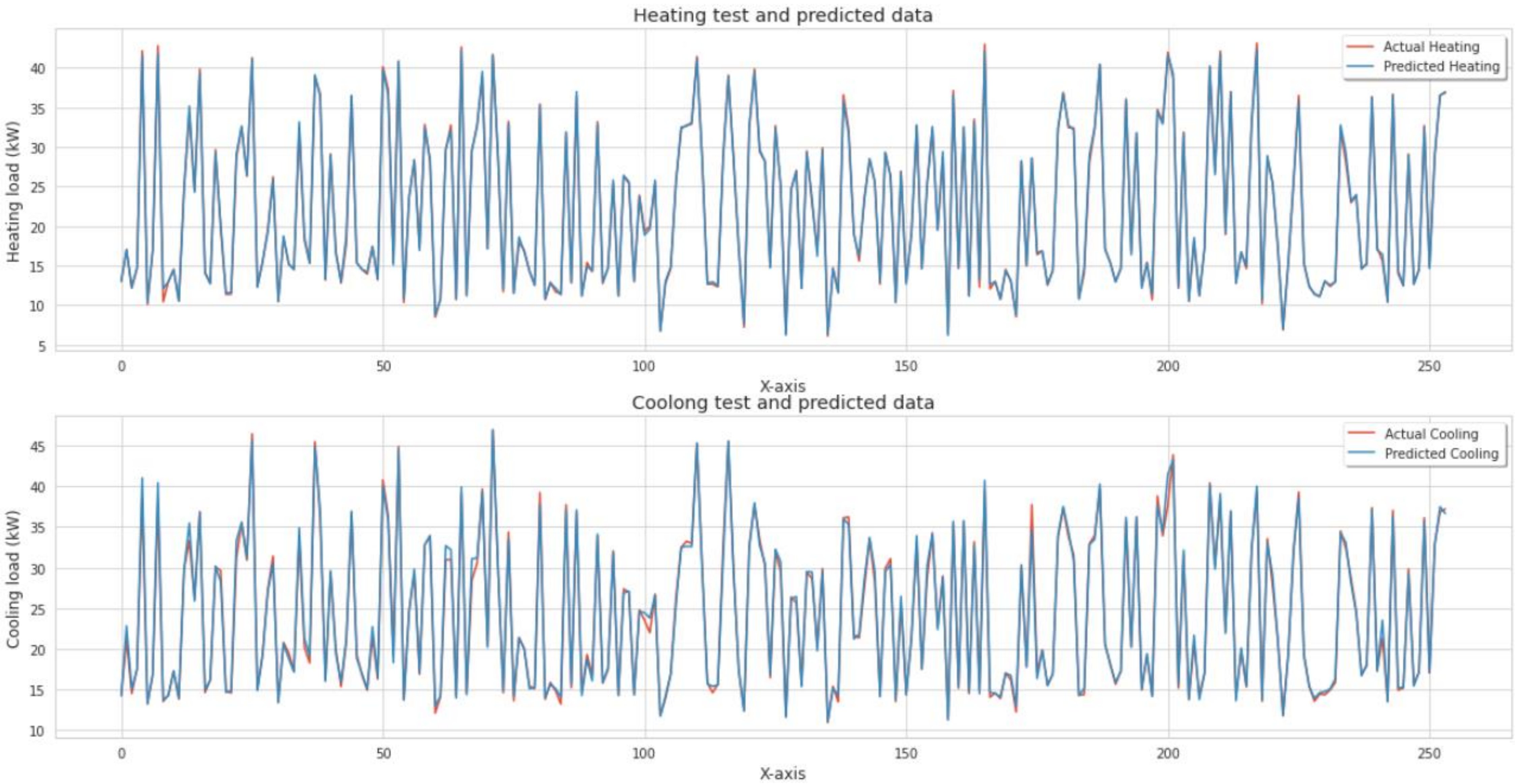
냉난방 부하 예측모델

```
[30] x_ax = range(len(y1_test))
     plt.figure(figsize=(20,10))
     plt.subplot(2,1,1)
     plt.plot(x_ax, y1_test, label="Actual Heating")
     plt.plot(x_ax, y1_pred, label="Predicted Heating")
     plt.title("Heating test and predicted data")
     plt.xlabel('X-axis')
     plt.ylabel('Heating load(kW)')
     plt.legend(loc='best',fancybox=True, shadow=True)
     plt.grid(True)

     plt.subplot(2,1,2)
     plt.plot(x_ax, y2_test, label="Actual Cooling")
     plt.plot(x_ax, y2_pred, label="Predicted Cooling")
     plt.title("Coolong test and predicted data")
     plt.xlabel('X-axis')
     plt.ylabel('Cooling load(kW)')
     plt.legend(loc='best',fancybox=True, shadow=True)
     plt.grid(True)

     plt.show()
```

냉난방 부하 예측모델



Heating test and predicted data
Actual Heating
Predicted Heating
Heating load (kW)
X-axis
Coolong test and predicted data
Actual Cooling
Predicted Cooling
Cooling load (kW)
X-axis

냉난방 부하 예측모델

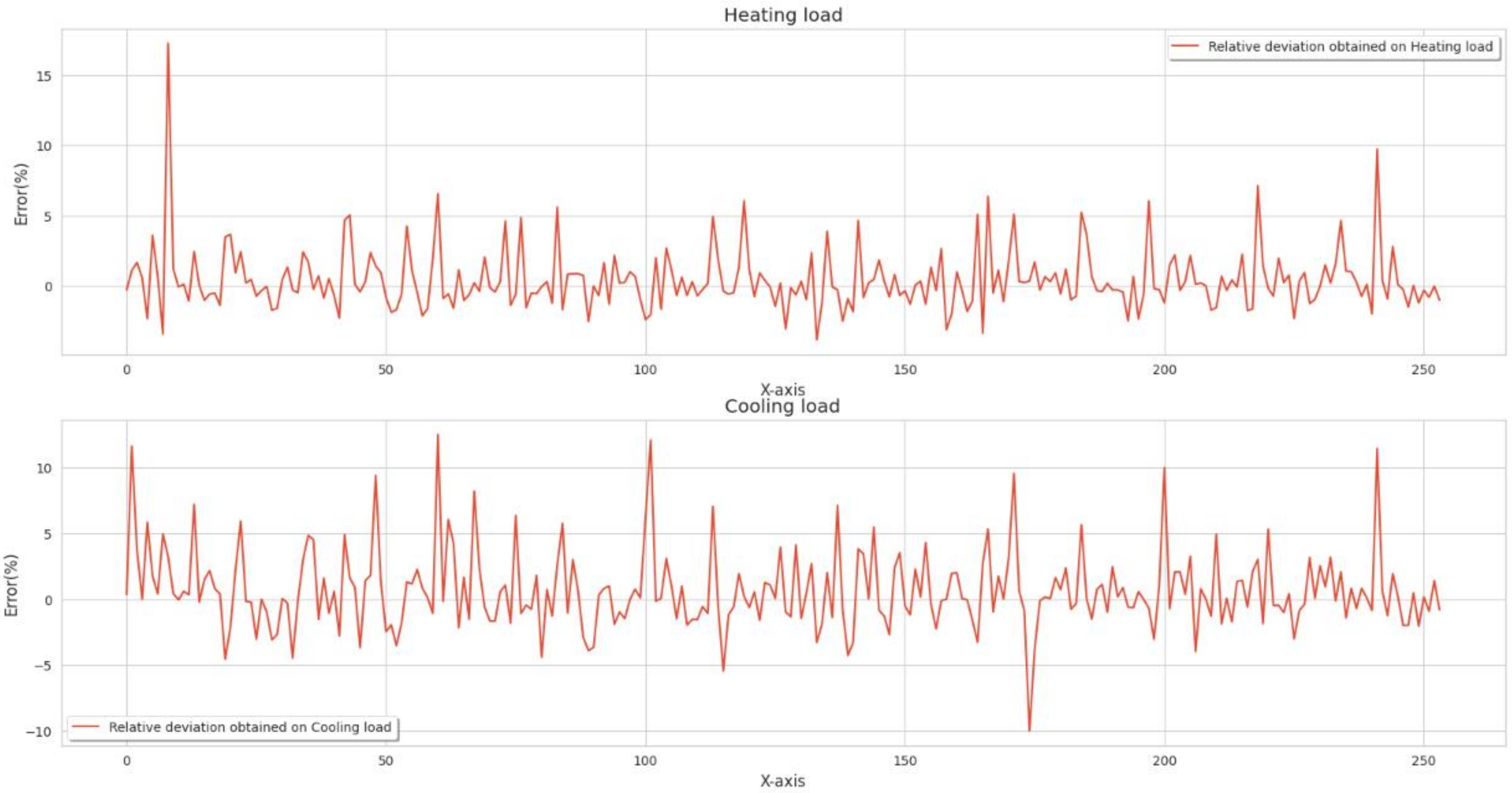
```
[31] def calculate_error_rate(y1_test, y1_pred):
         error_rate=[]
         for i in range(len(y1_pred)):
             error_rate.append((y1_pred[i] - y1_test.values[i])/y1_test.values[i]*100)
         return error_rate
```

```
[32] x_ax = range(len(y1_test))
     plt.figure(figsize=(20,10))
     plt.subplot(2,1,1)
     plt.plot(x_ax, calculate_error_rate(y1_test, y1_pred),
              label="Relative deviation obtained on Heating load")
     plt.title("Heating load")
     plt.xlabel('X-axis')
     plt.ylabel('Error(%)')
     plt.legend(loc='best',fancybox=True, shadow=True)
     plt.grid(True)

     plt.subplot(2,1,2)
     plt.plot(x_ax, calculate_error_rate(y2_test, y2_pred),
              label="Relative deviation obtained on Cooling load")
     plt.title("Cooling load")
     plt.xlabel('X-axis')
     plt.ylabel('Error(%)')
     plt.legend(loc='best',fancybox=True, shadow=True)
     plt.grid(True)

     plt.show()
```

냉난방 부하 예측모델



Heating load
Relative deviation obtained on Heating load
Error(%)
15
10
5
0
0
50
100
150
200
250
X-axis
Cooling load
Relative deviation obtained on Cooling load
Error(%)
10
5
0
−5
−10
0
50
100
150
200
250
X-axis

THANK YOU

kgpark88@gmail.com