```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\gaikw\appdata\
local\programs\python\python311\lib\site-packages (3.1.5)
Requirement already satisfied: et-xmlfile in c:\users\gaikw\appdata\
local\programs\python\python311\lib\site-packages (from openpyxl)
(2.0.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip available: 22.3 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

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

```
file_path = "ENB2012_data.xlsx"
data = pd.read_excel(file_path)
```

```
df = pd.DataFrame(data)
df.info()
df.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      768 non-null    float64
 1   X2      768 non-null    float64
 2   X3      768 non-null    float64
 3   X4      768 non-null    float64
 4   X5      768 non-null    float64
 5   X6      768 non-null    int64
 6   X7      768 non-null    float64
 7   X8      768 non-null    int64
 8   Y1      768 non-null    float64
 9   Y2      768 non-null    float64
dtypes: float64(8), int64(2)
memory usage: 60.1 KB
```

```
X1    0
X2    0
X3    0
X4    0
X5    0
X6    0
```

```
X7    0
X8    0
Y1    0
Y2    0
dtype: int64
```

```
df2 = df.dropna()
print(df2)
df2.isnull().sum()
```

```
       X1     X2     X3      X4   X5  X6   X7  X8     Y1     Y2
0    0.98  514.5  294.0  110.25  7.0   2  0.0   0  15.55  21.33
1    0.98  514.5  294.0  110.25  7.0   3  0.0   0  15.55  21.33
2    0.98  514.5  294.0  110.25  7.0   4  0.0   0  15.55  21.33
3    0.98  514.5  294.0  110.25  7.0   5  0.0   0  15.55  21.33
4    0.90  563.5  318.5  122.50  7.0   2  0.0   0  20.84  28.28
..    ...    ...    ...     ...  ...  ..  ...  ..    ...    ...
763  0.64  784.0  343.0  220.50  3.5   5  0.4   5  17.88  21.40
764  0.62  808.5  367.5  220.50  3.5   2  0.4   5  16.54  16.88
765  0.62  808.5  367.5  220.50  3.5   3  0.4   5  16.44  17.11
766  0.62  808.5  367.5  220.50  3.5   4  0.4   5  16.48  16.61
767  0.62  808.5  367.5  220.50  3.5   5  0.4   5  16.64  16.03

[768 rows x 10 columns]
```

```
X1    0
X2    0
X3    0
X4    0
X5    0
X6    0
X7    0
X8    0
Y1    0
Y2    0
dtype: int64
```

```
df2[df2.columns].corr()
```

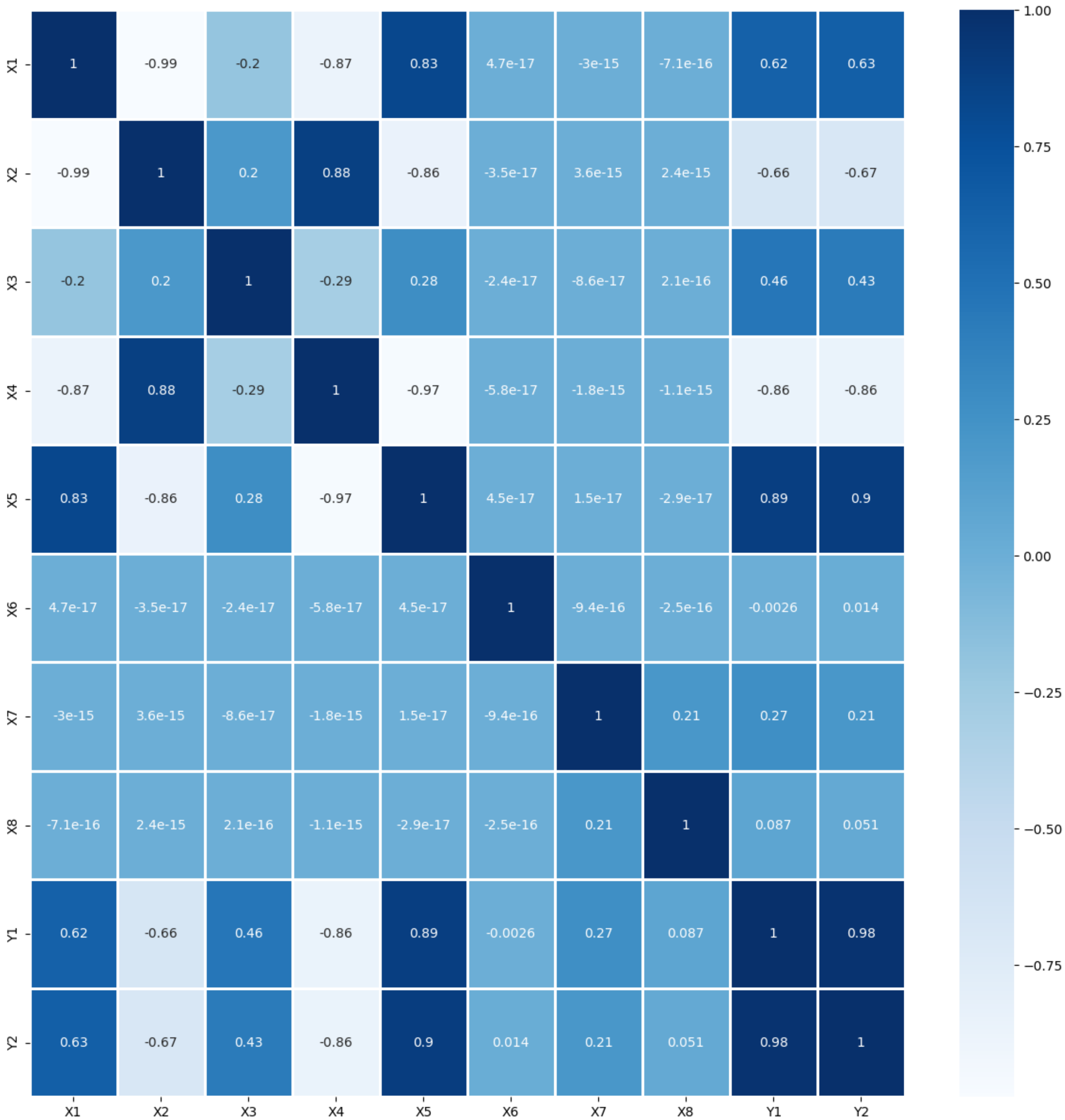
```
              X1            X2            X3            X4
X5  \
X1  1.000000e+00 -9.919015e-01 -2.037817e-01 -8.688234e-01  8.277473e-
01
X2 -9.919015e-01  1.000000e+00  1.955016e-01  8.807195e-01 -8.581477e-
01
X3 -2.037817e-01  1.955016e-01  1.000000e+00 -2.923165e-01  2.809757e-
01
X4 -8.688234e-01  8.807195e-01 -2.923165e-01  1.000000e+00 -9.725122e-
01
X5  8.277473e-01 -8.581477e-01  2.809757e-01 -9.725122e-01
```

```
1.000000e+00
X6  4.678592e-17 -3.459372e-17 -2.429499e-17 -5.830058e-17  4.492205e-
17
X7 -2.960552e-15  3.636925e-15 -8.567455e-17 -1.759011e-15  1.489134e-
17
X8 -7.107006e-16  2.438409e-15  2.067384e-16 -1.078071e-15 -2.920613e-
17
Y1  6.222719e-01 -6.581199e-01  4.556714e-01 -8.618281e-01  8.894305e-
01
Y2  6.343391e-01 -6.729989e-01  4.271170e-01 -8.625466e-01  8.957852e-
01

              X6            X7            X8        Y1        Y2
X1  4.678592e-17 -2.960552e-15 -7.107006e-16  0.622272  0.634339
X2 -3.459372e-17  3.636925e-15  2.438409e-15 -0.658120 -0.672999
X3 -2.429499e-17 -8.567455e-17  2.067384e-16  0.455671  0.427117
X4 -5.830058e-17 -1.759011e-15 -1.078071e-15 -0.861828 -0.862547
X5  4.492205e-17  1.489134e-17 -2.920613e-17  0.889430  0.895785
X6  1.000000e+00 -9.406007e-16 -2.549352e-16 -0.002587  0.014290
X7 -9.406007e-16  1.000000e+00  2.129642e-01  0.269842  0.207505
X8 -2.549352e-16  2.129642e-01  1.000000e+00  0.087368  0.050525
Y1 -2.586763e-03  2.698417e-01  8.736846e-02  1.000000  0.975862
Y2  1.428960e-02  2.075050e-01  5.052512e-02  0.975862  1.000000
```

```
plt.figure(figsize =(15,15))
sns.heatmap(data=df2[df2.columns].corr(), annot = True, linewidth = 2,
linecolor = 'white', cmap='Blues')
```

```
<Axes: >
```




```
from sklearn.model_selection import train_test_split
#Ignoring the features having weak/no correlation.
features_col = ['X1','X2','X4','X5','X7']
target_col = ['Y1']
x = df2[features_col]
y = df2[target_col]

x_train, x_test, y_train, y_test = train_test_split(x, y,
random_state=64, test_size=0.2)
print('X Train :\n', x_train.head())
```

```python
print('\nY Train :\n', y_train.head())
print('\nX Test :\n', x_test.head())
print('\nY Test :\n', y_test.head() ,'\n')


x_train = x_train.to_numpy()
y_train = y_train.to_numpy()
x_test = x_test.to_numpy()
y_test = y_test.to_numpy()

print('Shape of x_train :', x_train.shape)
print('Shape of y_train :', y_train.shape)
print('Shape of x_test :', x_test.shape)
print('Shape of y_test :', y_test.shape)
```

```
X Train :
        X1     X2     X4   X5    X7
588  0.82  612.5  147.0  7.0  0.40
128  0.69  735.0  220.5  3.5  0.10
443  0.86  588.0  147.0  7.0  0.25
644  0.76  661.5  122.5  7.0  0.40
127  0.71  710.5  220.5  3.5  0.10

Y Train :
          Y1
588  28.95
128  11.45
443  29.88
644  39.32
127  10.68

X Test :
        X1     X2     X4   X5    X7
270  0.71  710.5  220.5  3.5  0.10
445  0.82  612.5  147.0  7.0  0.25
93   0.62  808.5  220.5  3.5  0.10
670  0.62  808.5  220.5  3.5  0.40
236  0.62  808.5  220.5  3.5  0.10

Y Test :
          Y1
270  10.67
445  24.96
93   12.97
670  16.55
236  12.85

Shape of x_train : (614, 5)
Shape of y_train : (614, 1)
```

```
Shape of x_test : (154, 5)
Shape of y_test : (154, 1)
```

```
def multilinear_regression(x, y):
    b0 = np.ones(x.shape[0])
    new_x = np.concatenate((b0.reshape(-1,1), x), axis = 1)
    beta = np.linalg.inv(new_x.T @ new_x) @ new_x.T @ y
    return beta

beta = multilinear_regression(x_train, y_train)
beta
```

```
array([[ 8.95958970e+01],
       [-6.77105227e+01],
       [-2.94855410e-02],
       [-1.25763624e-01],
       [ 4.11175650e+00],
       [ 2.08259777e+01]])
```

```
def prediction(x, beta):
    b0 = np.ones(x.shape[0])
    new_x = np.concatenate((b0.reshape(-1,1), x), axis = 1)
    predicted_value = new_x @ beta
    return predicted_value

predicted_y1 = prediction(x_test, beta)

ss_res = np.sum((predicted_y1 - y_test)**2)
ss_tot = np.sum((predicted_y1 - y_test.mean())**2)
R_sq = 1 - (ss_res/ss_tot)
mse = np.mean((predicted_y1 - y_test)**2)
print('R-squared Value =', R_sq)
```

```
R-squared Value = 0.8910828666212713
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
LR = LinearRegression()
#Training
LR.fit(x_train, y_train)
#Predicting
y_pred_linear = LR.predict(x_test)

#Finding the goodness of fit
mse_linear = mean_squared_error(y_test, y_pred_linear)
R_sq_linear = r2_score(y_test, y_pred_linear)
print('Mean-squared Error =', mse_linear)
print('R-squared Value =', R_sq_linear)
```

```
Mean-squared Error = 10.93388128839549
R-squared Value = 0.8955413449874704
```

```python
from sklearn.linear_model import Ridge
ridge = Ridge(alpha = 0.2)
#Training
ridge.fit(x_train, y_train)
#Predicting
y_pred_ridge = ridge.predict(x_test)

#Finding the goodness of fit
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
R_sq_ridge = r2_score(y_test, y_pred_ridge)
print('Mean-squared Error =', mse_ridge)
print('R-squared Value =', R_sq_ridge)
```

```
Mean-squared Error = 11.018535552002326
R-squared Value = 0.8947325863880121
```

```python
from sklearn.linear_model import Lasso
lasso = Lasso(alpha = 0.2)
#Training
lasso.fit(x_train, y_train)
#Predicting
y_pred_lasso = lasso.predict(x_test)

#Finding the goodness of fit
mse_lasso = mean_squared_error(y_test, y_pred_lasso)
R_sq_lasso = r2_score(y_test, y_pred_lasso)
print('Mean-squared Error =', mse_lasso)
print('R-squared Value =', R_sq_lasso)
```

```
Mean-squared Error = 12.64835459163122
R-squared Value = 0.8791618388828112
```

```python
data = {
    'Metrics' : ['R2 Score', 'MSE', 'RMSE'],
    'Scratch' : [R_sq, mse, (mse)**0.5],
    'sklearn' : [R_sq_linear, mse_linear, (mse_linear)**0.5],
    'Ridge' : [R_sq_ridge, mse_ridge, (mse_ridge)**0.5],
    'Lasso' : [R_sq_lasso, mse_ridge, (mse_lasso)**0.5]
}
data = pd.DataFrame(data)
data = data.set_index('Metrics')
data
```

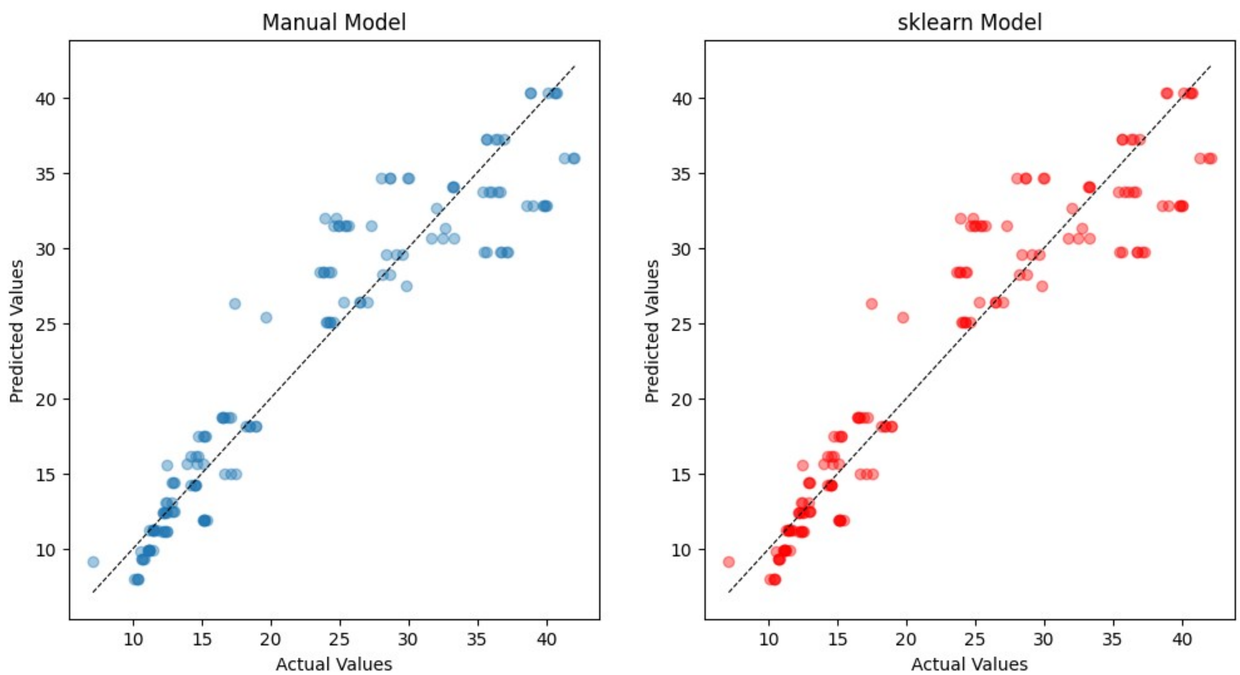
Metrics	Scratch	sklearn	Ridge	Lasso
R2 Score	0.891083	0.895541	0.894733	0.879162
MSE	10.933881	10.933881	11.018536	11.018536
RMSE	3.306642	3.306642	3.319418	3.556453

```python
plt.figure(figsize=(12, 6))
```

```
# Manual Model
plt.subplot(1,2,1)
plt.scatter(y_test, predicted_y1, alpha = 0.4)
plt.title('Manual Model')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=0.8)


# sklearn Model
plt.subplot(1,2,2)
plt.scatter(y_test, y_pred_linear, alpha = 0.4, c='red')
plt.title('sklearn Model')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=0.8)
```

```
[<matplotlib.lines.Line2D at 0x12ae79a4c10>]
```




```
correlation = df2.corrwith(df2['Y1'])
correlation
```

```
X1    0.622272
X2   -0.658120
X3    0.455671
X4   -0.861828
X5    0.889430
```

```
X6   -0.002587
X7    0.269842
X8    0.087368
Y1    1.000000
Y2    0.975862
dtype: float64
```

Recommendations

Insulation : apparantely has a positive impact on heating load, so more the insulation less the heating demand.

Relative Compactness: Large negative coefficient (-2.549774e+01) indicates that more compact buildings significantly reduce heating load. Suggestion: Design compact structures to minimize exposed surface area and maximize energy efficiency.

Surface Area: Large positive coefficient (1.139458e+15) indicates that increasing surface area dramatically increases the heating load. Suggestion: Optimize the surface area to reduce heat loss. Avoid designs with unnecessarily large exposed areas. Use materials with better insulation properties.

X3 should be optimized for better energy performance.

```
```