Lasso_Ridge_Class_Assignment

February 16, 2025

1 Ridge And Lasso Regression

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, r2_score, mean_squared_error
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
from sklearn.model_selection import GridSearchCV
```

[2]:
```
df = pd.read_csv('Advertising.csv')
df.head()
```

[2]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

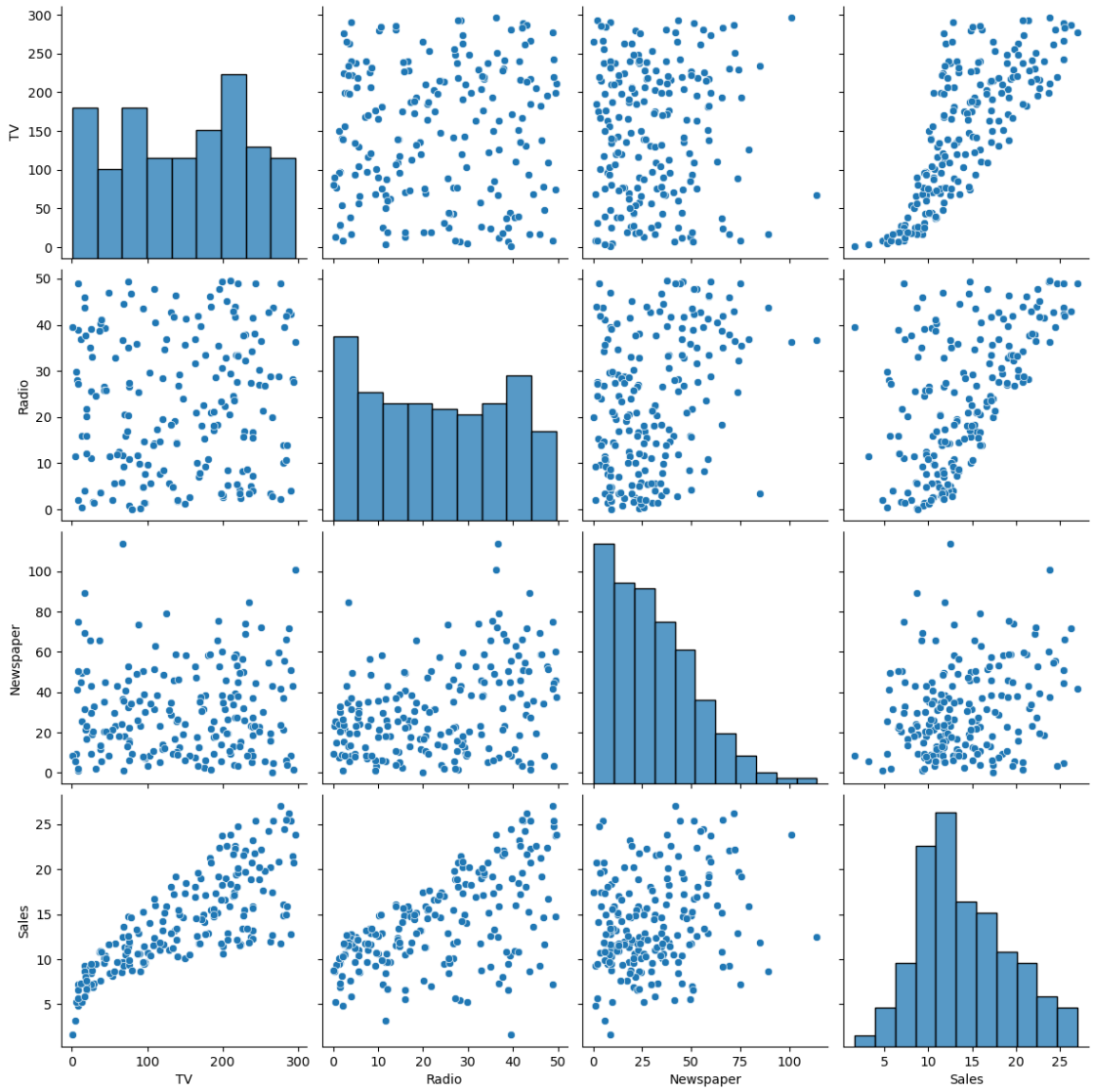
[3]:
```
df = df.iloc[ : , 1: ]
df.head()
```

[3]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

[4]:
```
sns.pairplot(data = df , height=3)
```

[4]:
```
<seaborn.axisgrid.PairGrid at 0x2b028cd7350>
```



[5]:
```
x = df[['TV','Radio','Newspaper']]
y = df['Sales']
```

[6]:
```
x.head()
```

[6]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

[7]:
```
y.head()
```

[7]:
```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: Sales, dtype: float64
```

[8]:
```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.
↪20,random_state=42)
```

[9]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
def calc_vif(X):
    vif = pd.DataFrame()
    vif["VIF"] = [variance_inflation_factor(X.values,i) for i in range(X.
↪shape[1])]
    return(vif)
```

[10]:
```
X = df.iloc[:,:-1]
calc_vif(X)
```

[10]:
```
        VIF
0  2.486772
1  3.285462
2  3.055245
```

[11]:
```
alpha_values = np.linspace(-3, 3, 10)
print(alpha_values)
```

```
[-3.         -2.33333333 -1.66666667 -1.         -0.33333333  0.33333333
  1.          1.66666667  2.33333333  3.        ]
```

[12]:
```
from sklearn.linear_model import Ridge
# Ridge Regression with hyperparameter tuning
ridge = Ridge()
# Define the hyperparameter grid for alpha values (regularization strength)
alpha_values = { 'alpha': np.logspace(-3, 3, 10)} # 10 values from 10^-3 to 10^3

ridge_cv = GridSearchCV(ridge, alpha_values, cv=5,␣
↪scoring='neg_mean_squared_error')
ridge_cv.fit(X_train, y_train) # Train Ridge regression model with␣
↪cross-validation

GridSearchCV(cv=5, estimator=Ridge(), param_grid={'alpha': np.logspace(-3, 3,␣
↪10)}, scoring='neg_mean_squared_error')
```

[12]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': array([1.00000000e-03, 4.64158883e-03,
2.15443469e-02, 1.00000000e-01,
```

```
       4.64158883e-01, 2.15443469e+00, 1.00000000e+01, 4.64158883e+01,
       2.15443469e+02, 1.00000000e+03])},
             scoring='neg_mean_squared_error')
```

[13]:
```
# Best Ridge Model
best_ridge = ridge_cv.best_estimator_
y_pred_ridge = best_ridge.predict(X_test) # Predictions on test data
```

[14]:
```
# Ridge RMSE
ridge_rmse = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
r2_ridge = r2_score(y_test,y_pred_ridge)
print("Best ridge Alpha :",ridge_cv.best_params_['alpha'])
print("RMSE :", ridge_rmse)
print("R2 :",r2_ridge)
```

```
Best ridge Alpha : 0.001
RMSE : 1.7815996608176399
R2 : 0.8994380241817195
```

[15]:
```
from sklearn.linear_model import Lasso
# Lasso Regression with hyperparameter tuning
lasso = Lasso()
lasso_cv = GridSearchCV(lasso, alpha_values, cv=5,␣
 ↪scoring='neg_mean_squared_error')
# Define the hyperparameter grid for alpha values (regularization strength)
lasso_cv.fit(X_train, y_train) # Train Lasso regression model with␣
 ↪cross-validation
GridSearchCV(cv=5, estimator=Lasso(), param_grid={'alpha': np.logspace(-3, -3,␣
 ↪10)}, scoring='neg_mean_squared_error')
```

[15]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': array([0.001, 0.001, 0.001, 0.001, 0.001,
0.001, 0.001, 0.001, 0.001,
       0.001])},
             scoring='neg_mean_squared_error')
```

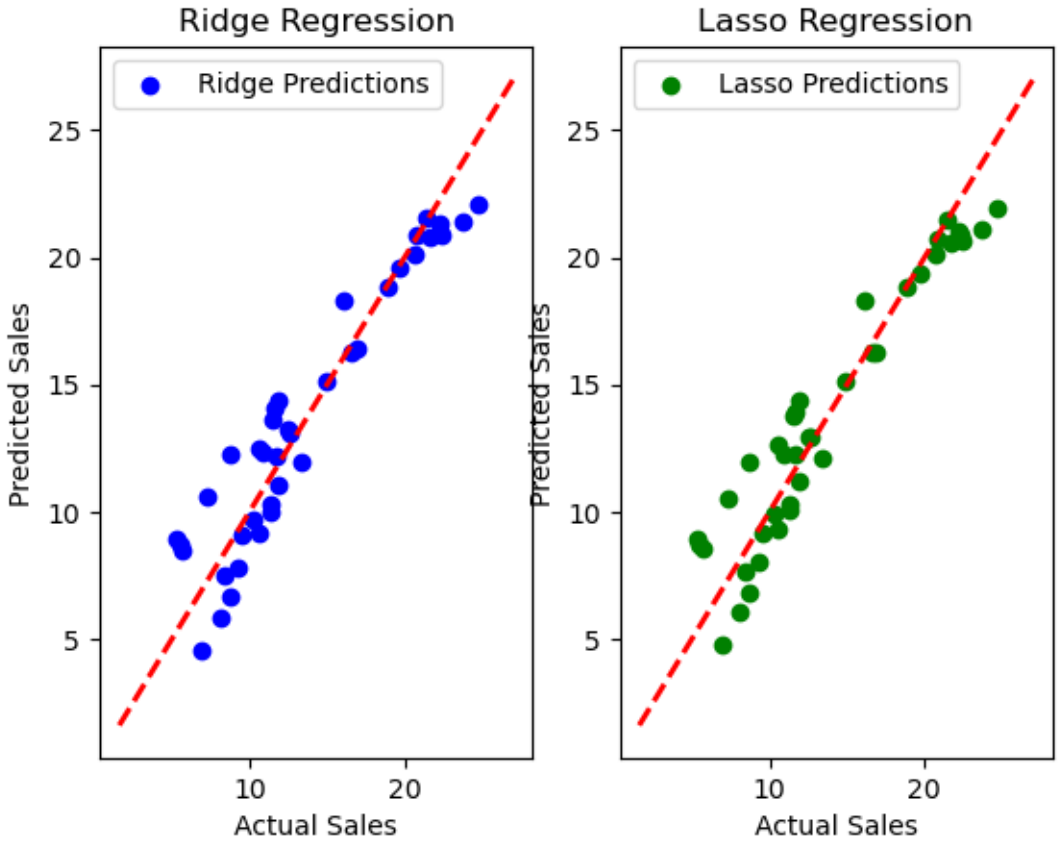
[16]:
```
# Best Lasso Model
best_lasso = lasso_cv.best_estimator_
y_pred_lasso = best_lasso.predict(X_test) # Predictions on test data
```

[17]:
```
# Lasso RMSE
lasso_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
r2_lasso = r2_score(y_test,y_pred_lasso)
print("Best Lasso Alpha :",lasso_cv.best_params_['alpha'])
print("RMSE :", lasso_rmse)
print("R2 :",r2_lasso)
```

```
Best Lasso Alpha : 2.154434690031882
```

```
RMSE : 1.7677097690307533
R2 : 0.9009999351697155
```

[18]:
```
# Ridge Regression Plot
plt.subplot(1,2,1)
plt.scatter(y_test,y_pred_ridge,color='blue',label="Ridge Predictions")
plt.plot([y.min(),y.max()],[y.min(),y.max()],'r--',lw=2)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Ridge Regression')
plt.legend()

# Lasso Regression Plot
plt.subplot(1,2,2)
plt.scatter(y_test,y_pred_lasso,color='green',label="Lasso Predictions")
plt.plot([y.min(),y.max()],[y.min(),y.max()],'r--',lw=2)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Lasso Regression')
plt.legend()
plt.show()
```



[19]:
```python
# Print the coefficients of the lasso with feature names
print("Lasso Coefficients :")
for feature, coef in zip(x.columns, best_lasso.coef_):
    print(f"feature : {feature}, Coefficient , {coef} ")
print("Ridge Coefficients :")
for feature, coef in zip(x.columns, best_lasso.coef_):
    print(f"feature : {feature}, Coefficient , {coef} ")
```

```
Lasso Coefficients :
feature : TV, Coefficient , 0.044516937884577404
feature : Radio, Coefficient , 0.1808414992503233
feature : Newspaper, Coefficient , 0.0
Ridge Coefficients :
feature : TV, Coefficient , 0.044516937884577404
feature : Radio, Coefficient , 0.1808414992503233
feature : Newspaper, Coefficient , 0.0
```

Practice Assignment

February 16, 2025

0.1 Practice Dataset

[67]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

Importing the dataframe

[68]:
```
df = pd.read_excel('Practice dataset.xlsx')
df.head()
```

[68]:

	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

Metadata

[69]:
```
# All the information about the dataset
df.info()
```

```
<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
```

[70]:
```
# Checking for missing values
df.isnull().sum()
```

[70]:
```
X1    0
X2    0
X3    0
X4    0
X5    0
X6    0
X7    0
X8    0
Y1    0
Y2    0
dtype: int64
```

[71]:
```
# Checking for duplicate values
df.duplicated().sum()
```

[71]:
```
0
```

[72]:
```
# Descriptive statistics
df.describe()
```

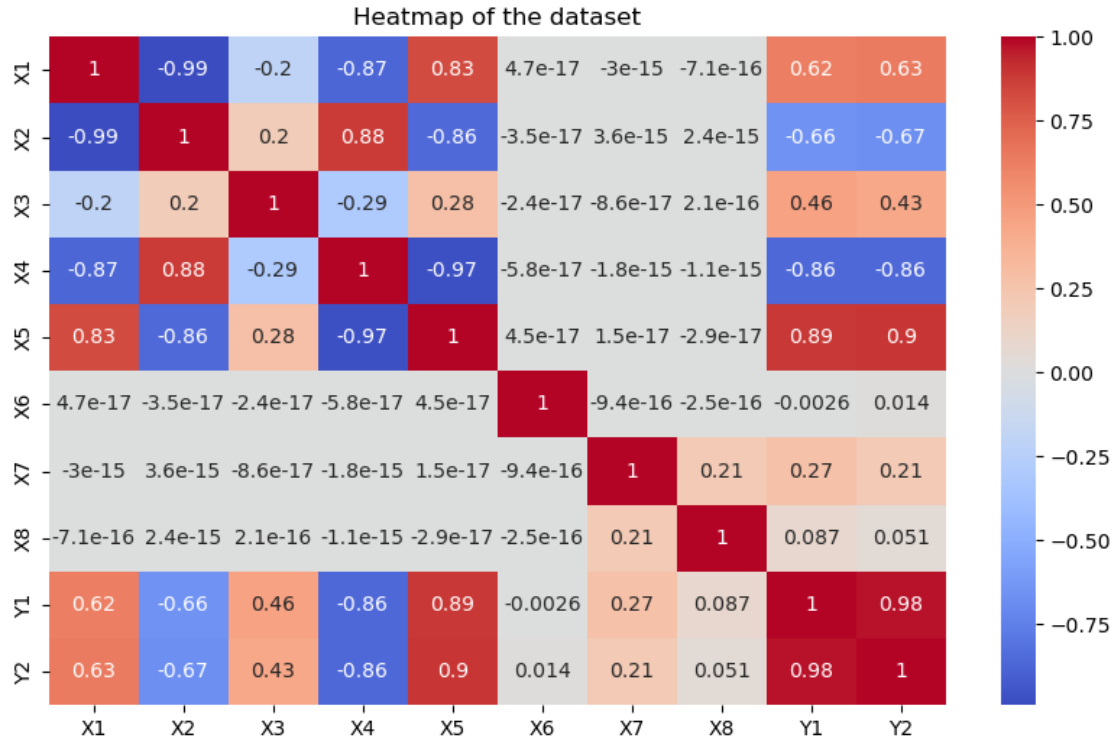
[72]:

	X1	X2	X3	X4	X5	X6	\
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	

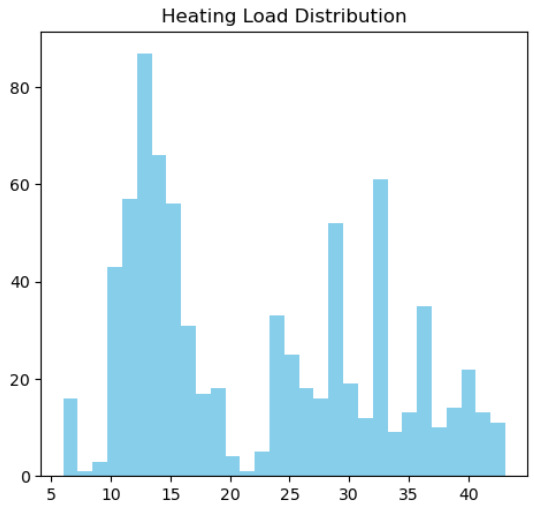
	X7	X8	Y1	Y2
count	768.000000	768.00000	768.000000	768.000000
mean	0.234375	2.81250	22.307195	24.587760
std	0.133221	1.55096	10.090204	9.513306
min	0.000000	0.00000	6.010000	10.900000
25%	0.100000	1.75000	12.992500	15.620000
50%	0.250000	3.00000	18.950000	22.080000
75%	0.400000	4.00000	31.667500	33.132500
max	0.400000	5.00000	43.100000	48.030000

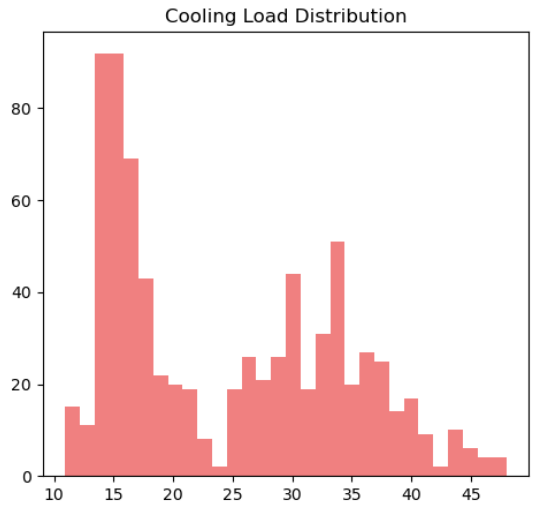
Data Visualization

[73]:
```
# heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Heatmap of the dataset')
plt.show()
```



[74]:
```
# Visualize the distribution of target variables
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.hist(df['Y1'], bins=30, color='skyblue')
plt.title('Heating Load Distribution')
plt.subplot(1, 2, 2)
plt.hist(df['Y2'], bins=30, color='lightcoral')
plt.title('Cooling Load Distribution')
plt.show()
```





[75]:
```
df.head()
```

[75]:

	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

[76]:
```
x = df.drop(['Y1','Y2'], axis=1)
y = df.drop(['X1','X2','X3','X4','X5','X6','X7','X8'], axis=1)
```

[77]:
```
y_heating = y['Y1']
y_cooling = y['Y2']
```

Ridge Regression for Heating Load

[78]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y_heating, test_size=0.
 ↪2, random_state=0)
```

[79]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
# Ridge Regression with hyperparameter tuning
ridge = Ridge()
# Define the hyperparameter grid for alpha values (regularization strength)
alpha_values = { 'alpha': np.logspace(-3, 3, 10)} # 10 values from 10^-3 to 10^3

ridge_cv = GridSearchCV(ridge, alpha_values, cv=5,␣
 ↪scoring='neg_mean_squared_error')
```

```
ridge_cv.fit(x_train, y_train) # Train Ridge regression model with␣
 ↪cross-validation

GridSearchCV(cv=5, estimator=Ridge(), param_grid={'alpha': np.logspace(-3, 3,␣
 ↪10)}, scoring='neg_mean_squared_error')


# Best Ridge Model
best_ridge = ridge_cv.best_estimator_
y_pred_ridge = best_ridge.predict(x_test) # Predictions on test data


# Ridge RMSE
ridge_rmse = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
r2_ridge = r2_score(y_test,y_pred_ridge)
print("Best ridge Alpha :",ridge_cv.best_params_['alpha'])
print("RMSE :", ridge_rmse)
print("R2 :",r2_ridge)
```

```
Best ridge Alpha : 0.001
RMSE : 3.1788282595545687
R2 : 0.9084914397742132
```

Lasso Regression for Heating Load

[80]:
```
from sklearn.linear_model import Lasso
# Lasso Regression with hyperparameter tuning
lasso = Lasso()
lasso_cv = GridSearchCV(lasso, alpha_values, cv=5,␣
 ↪scoring='neg_mean_squared_error')
# Define the hyperparameter grid for alpha values (regularization strength)
lasso_cv.fit(x_train, y_train) # Train Lasso regression model with␣
 ↪cross-validation
GridSearchCV(cv=5, estimator=Lasso(), param_grid={'alpha': np.logspace(-3, -3,␣
 ↪10)}, scoring='neg_mean_squared_error')


# Best Lasso Model
best_lasso = lasso_cv.best_estimator_
y_pred_lasso = best_lasso.predict(x_test) # Predictions on test data



# Lasso RMSE
lasso_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
r2_lasso = r2_score(y_test,y_pred_lasso)
print("Best Lasso Alpha :",lasso_cv.best_params_['alpha'])
print("RMSE :", lasso_rmse)
```

```
print("R2 :",r2_lasso)
```

```
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 4.781e+01, tolerance: 5.007e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.635e+02, tolerance: 4.787e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.309e+02, tolerance: 4.945e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.676e+02, tolerance: 4.790e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 4.681e+01, tolerance: 4.875e+00
  model = cd_fast.enet_coordinate_descent(
```

```
Best Lasso Alpha : 0.001
RMSE : 3.1916005738152298
R2 : 0.9077546122237674
```

```
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.258e+02, tolerance: 6.103e+00
  model = cd_fast.enet_coordinate_descent(
```

Evaluating Heating of both models using mean square error

[81]:
```
from sklearn.metrics import mean_squared_error

y_pred_ridge = ridge_cv.predict(x_test)
```

```
y_pred_lasso = lasso_cv.predict(x_test)

mse_ridge = mean_squared_error(y_test, y_pred_ridge)
mse_lasso = mean_squared_error(y_test, y_pred_lasso)

print("Ridge MSE:", mse_ridge)
print("Lasso MSE:", mse_lasso)
```

```
Ridge MSE: 10.104949103742728
Lasso MSE: 10.186314222777705
```

Ridge Regression for Cooling Load

[82]:
```
from sklearn.model_selection import train_test_split
x_train1, x_test1, y_train1, y_test1 = train_test_split(x, y_cooling,␣
 ↪test_size=0.2, random_state=0)
```

[83]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
# Ridge Regression with hyperparameter tuning
ridge1 = Ridge()
# Define the hyperparameter grid for alpha values (regularization strength)
alpha_values = { 'alpha': np.logspace(-3, 3, 10)} # 10 values from 10^-3 to 10^3

ridge_cv1 = GridSearchCV(ridge1, alpha_values, cv=5,␣
 ↪scoring='neg_mean_squared_error')
ridge_cv1.fit(x_train1, y_train1) # Train Ridge regression model with␣
 ↪cross-validation

GridSearchCV(cv=5, estimator=Ridge(), param_grid={'alpha': np.logspace(-3, 3,␣
 ↪10)}, scoring='neg_mean_squared_error')


# Best Ridge Model
best_ridge1 = ridge_cv1.best_estimator_
y_pred_ridge1 = best_ridge1.predict(x_test1) # Predictions on test data


# Ridge RMSE
ridge_rmse1 = np.sqrt(mean_squared_error(y_test1, y_pred_ridge1))
r2_ridge1 = r2_score(y_test1,y_pred_ridge1)
print("Best ridge Alpha :",ridge_cv1.best_params_['alpha'])
print("RMSE :", ridge_rmse1)
print("R2 :",r2_ridge1)
```

```
Best ridge Alpha : 0.001
RMSE : 3.272220047068651
R2 : 0.8861585541962702
```

Lasso Regression for Cooling Load

[84]:
```
from sklearn.linear_model import Lasso
# Lasso Regression with hyperparameter tuning
lasso1 = Lasso()
lasso_cv1 = GridSearchCV(lasso1, alpha_values, cv=5,␣
 ↪scoring='neg_mean_squared_error')
# Define the hyperparameter grid for alpha values (regularization strength)
lasso_cv1.fit(x_train1, y_train1) # Train Lasso regression model with␣
 ↪cross-validation
GridSearchCV(cv=5, estimator=Lasso(), param_grid={'alpha': np.logspace(-3, -3,␣
 ↪10)}, scoring='neg_mean_squared_error')


# Best Lasso Model
best_lasso1 = lasso_cv1.best_estimator_
y_pred_lasso1 = best_lasso1.predict(x_test) # Predictions on test data



# Lasso RMSE
lasso_rmse1 = np.sqrt(mean_squared_error(y_test1, y_pred_lasso1))
r2_lasso1 = r2_score(y_test1,y_pred_lasso1)
print("Best Lasso Alpha :",lasso_cv1.best_params_['alpha'])
print("RMSE :", lasso_rmse1)
print("R2 :",r2_lasso1)
```

```
Best Lasso Alpha :
```

```
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 4.519e+01, tolerance: 4.560e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 2.971e+02, tolerance: 4.315e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 8.898e+01, tolerance: 4.435e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
```

```
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 2.276e+02, tolerance: 4.253e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 9.282e+00, tolerance: 4.392e+00
  model = cd_fast.enet_coordinate_descent(
c:\Users\Neil\anaconda3\Lib\site-
packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.340e+02, tolerance: 5.492e+00
  model = cd_fast.enet_coordinate_descent(

 0.001
RMSE : 3.281730256047722
R2 : 0.8854958669614633
```

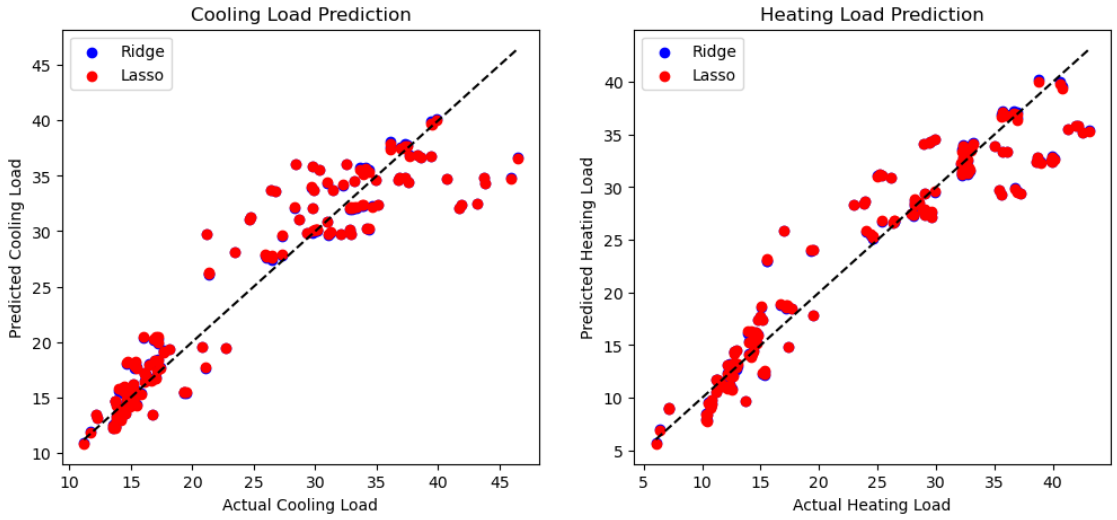
Evaluating Cooling of both models using mean square error

[85]:
```
from sklearn.metrics import mean_squared_error

y_pred_ridge1 = ridge_cv1.predict(x_test)
y_pred_lasso1 = lasso_cv1.predict(x_test)

mse_ridge1 = mean_squared_error(y_test1, y_pred_ridge1)
mse_lasso1 = mean_squared_error(y_test1, y_pred_lasso1)

print("Ridge MSE:", mse_ridge1)
print("Lasso MSE:", mse_lasso1)
```

```
Ridge MSE: 10.707424036437965
Lasso MSE: 10.769753473459048
```

[86]:
```
 # Ridge vs Lasso Comparison for Cooling Load
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(y_test1, y_pred_ridge1, color='blue', label='Ridge')
plt.scatter(y_test1, y_pred_lasso1, color='red', label='Lasso')
plt.plot([min(y_test1), max(y_test1)], [min(y_test1), max(y_test1)],␣
 ↪color='black', linestyle='--')
plt.xlabel('Actual Cooling Load')
plt.ylabel('Predicted Cooling Load')
plt.title('Cooling Load Prediction')
plt.legend()
```

```
# Ridge vs Lasso Comparison for Heating Load
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_ridge, color='blue', label='Ridge')
plt.scatter(y_test, y_pred_lasso, color='red', label='Lasso')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black',␣
 ↪linestyle='--')
plt.xlabel('Actual Heating Load')
plt.ylabel('Predicted Heating Load')
plt.title('Heating Load Prediction')
plt.legend()
plt.show()
```



[91]:
```
# Extract coefficients from both Lasso models
lasso_coefs = best_lasso.coef_
lasso1_coefs = best_lasso1.coef_

# Create a DataFrame with the coefficients
coefs_df = pd.DataFrame([lasso_coefs, lasso1_coefs], index=['Lasso Model for␣
 ↪Heating', 'Lasso Model for Cooling'], columns=x.columns)
coefs_df
```

[91]:

	X1	X2	X3	X4	X5	X6	X7	X8
Lasso Model for Heating	-48.123597	-0.060907	0.054349	-0.010020	4.242992	-0.018314	19.757459	0.160407
Lasso Model for Cooling	-55.297681	-0.061855	0.037394	-0.006894	4.508389	0.144312	15.015196	0.024498

0.1.1 Interpretation of Lasso Regression Coefficients

The table below shows the coefficients estimated by two Lasso regression models—one for predicting **Heating Load** and the other for **Cooling Load**—based on eight features (X1 to X8).

Feature	Lasso Model for Heating	Lasso Model for Cooling
X1	-48.123597	-55.297681
X2	-0.060907	-0.061855
X3	0.054349	0.037394
X4	-0.010020	-0.006894
X5	4.242992	4.508389
X6	-0.018314	0.144312
X7	19.757459	15.015196
X8	0.160407	0.024498

Key Points:

- **Magnitude & Impact:**
  - **X1 (Relative Compactness):**
    - **Heating:** A coefficient of -48.12 indicates that for each one-unit increase in X1, the Heating Load decreases by about 48.12 units, holding all else constant.
    - **Cooling:** Similarly, a coefficient of -55.30 suggests a stronger negative impact on Cooling Load.
  - **X5 (Overall Height):**
    - Positive coefficients in both models (4.24 for heating and 4.51 for cooling) indicate that as the overall height increases, both heating and cooling loads tend to increase.
  - **X7 (Glazing Area):**
    - With coefficients of 19.76 (heating) and 15.02 (cooling), this feature shows a considerable positive influence on the target variables, though slightly more on Heating Load.
- **Direction of Effect (Sign of Coefficients):**
  - **Negative Coefficients (e.g., X1, X2, X4):**
    - Indicate an inverse relationship. For instance, higher values of X1 are associated with lower energy loads.
  - **Positive Coefficients (e.g., X3, X5, X7, X8):**
    - Suggest a direct relationship, where increases in these features lead to higher energy loads.
- **Differences Between Models:**
  - **X6 (Orientation):**
    - For Heating Load, the coefficient is slightly negative (-0.0183), suggesting a minor decrease in heating load with an increase in orientation value.
    - For Cooling Load, the coefficient is positive (0.1443), indicating that a higher orientation value could slightly increase the cooling load.
  - **X8 (Glazing Area Distribution):**
    - A modest positive impact on Heating Load (0.1604) but almost negligible for Cooling Load (0.0245).
- **Lasso Regularization:**

  - Lasso tends to shrink coefficients towards zero, and it can even set some coefficients exactly to zero if a feature is not useful. In this case, none of the coefficients are zero, which implies that all features are contributing to the predictions for both Heating and Cooling Loads, albeit with different levels of influence.

Overall Implications:

- **Relative Importance:**
  - **X1** and **X7** have large absolute coefficients, suggesting they are among the most influential features in predicting both Heating and Cooling Loads.
- **Model Sensitivity:**
  - The differences in coefficient signs and magnitudes (especially for X6) hint that the relationship between features and the target variables may differ between heating and cooling scenarios. This reinforces the need for tailored models or further feature engineering when predicting these energy loads.

0.1.2 Model Performance Evaluation Using Mean Squared Error (MSE)

We evaluate the performance of Ridge and Lasso Regression models on two targets—Heating Load and Cooling Load—using MSE. The obtained MSE values are as follows:

Heating Load Prediction:

- **Ridge Regression MSE:** 10.104949103742728
- **Lasso Regression MSE:** 10.186314222777705

Cooling Load Prediction:

- **Ridge Regression MSE:** 10.707424036437965
- **Lasso Regression MSE:** 10.769753473459048

Interpretation:

1. **Heating Load Predictions:**
   - **Ridge Regression** achieves an MSE of approximately **10.105**, which is slightly lower than the **10.186** obtained by **Lasso Regression**.
   - This indicates that the Ridge model is marginally better at predicting Heating Load, as its predictions are on average closer to the actual values.
2. **Cooling Load Predictions:**
   - For Cooling Load, **Ridge Regression** records an MSE of about **10.707**, compared to **10.770** for **Lasso Regression**.
   - Again, the Ridge model shows a slight edge in performance over the Lasso model.
3. **Overall Comparison:**
   - **Ridge Regression** consistently demonstrates lower MSE values for both Heating and Cooling Load predictions.

- The differences in MSE are relatively small, suggesting that both models are performing comparably. However, the slight advantage in Ridge's performance might make it the preferred choice if minimal improvement is desired.

4. **Conclusion:**
   - **MSE** is a measure of the average squared difference between the predicted and actual values. A lower MSE is indicative of better predictive accuracy.
   - Given the marginally lower MSE values for Ridge Regression across both tasks, it can be inferred that Ridge Regression might capture the underlying patterns in the data a bit more effectively than Lasso Regression for this particular dataset.

Experiment/Practical 4 Ridge and Lasso Regression

Title: Implementation of Ridge and Lasso Regression

Aim: To apply Ridge and Lasso regression algorithms for prediction and model regularization

Objective: Students will learn

- Implementation of Ridge and Lasso regression algorithms on the given dataset(s).
- To compare and contrast both algorithms' performance and understand their impact on model regularization.
- To visualize and interpret the results effectively.

Problem statement

Use the given datasets to demonstrate Ridge and Lasso regression, predicting a dependent variable based on independent variables while applying regularization to prevent overfitting.

Explanation/Stepwise Procedure/ Algorithm:

- Give a brief description of Ridge and Lasso regression.
  - **Ridge Regression:**
    - Ridge regression (L2 regularization) is a linear model that adds a penalty equal to the sum of the squared coefficients to the loss function. This regularization technique reduces model complexity and prevents overfitting by shrinking the coefficients. Unlike Lasso, Ridge does not eliminate coefficients but brings them closer to zero.
    - It is particularly useful when dealing with multicollinearity (high correlation among input features).
  - **Lasso Regression:**
    - Lasso regression (L1 regularization) adds a penalty equal to the absolute value of the coefficients to the loss function. It performs both regularization and feature selection by driving some coefficients to

exactly zero. This makes Lasso effective for sparse models with fewer predictors.
- It is useful when there are many features but only a few are expected to be important.

- Give mathematical formulation of Ridge and Lasso regression
  - Ridge(l2):

$$J(\beta) = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2$$

  - Where J(Beta) = COST Funtion and Lambda * summation(beta(j) ^2) is the regularization term.

  Lasso(L1):

$$J(\beta) = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

  - Where J(Beta) = COST Funtion and Lambda * summation(beta(j)) is the regularization term.

- Write the importance of Ridge and Lasso regression in data analysis.

  - **Ridge Regression:**

    - Handles multicollinearity by adding bias to the regression estimates, leading to more reliable and stable predictions.
    - Helps in controlling model complexity and reducing overfitting, especially when the number of predictors is large.

  - **Lasso Regression:**

    - Performs automatic feature selection by setting irrelevant feature coefficients to zero.
    - Useful for high-dimensional data where feature selection is necessary for model interpretability and efficiency.
    - Enhances model generalization by reducing overfitting through sparse solutions.

- Mention applications of Ridge and Lasso regression in real-world scenarios.

  - **Ridge Regression:**

    - **Financial Forecasting:** Predicting stock prices, sales, or economic indicators where features are highly correlated.
    - **Medical Research:** Predicting patient outcomes using a large number of clinical variables with multicollinearity.
    - **Climate Modeling:** Analyzing environmental data with correlated atmospheric features.

  - **Lasso Regression:**

    - **Genomics and Bioinformatics:** Identifying relevant genes influencing diseases from a vast number of genetic features.
    - **Marketing Analytics:** Selecting significant variables that influence customer purchase behavior.
    - **Image Processing:** Sparse representation and feature selection in high-dimensional image data.
    - **Text Classification:** Selecting important words (features) in Natural Language Processing (NLP) tasks.

- Brief explanation of performance metrics (e.g., R^2, Mean Squared Error, Root Mean Squared Error).

  Performance metrics evaluate how well the model fits on the test set

  R^2 measures how well the independent variable(s) explain the variance in the dependent variable. R^2 ranges from 0 to 1, where a value closer to 1 indicates that the model explains a large proportion of the variance, while a value near 0 means the model doesn't explain much of the variation.

  MSE is the average of the squared differences between predicted and actual values. It gives an idea of how far off predictions are from actual values. A lower MSE indicates better model performance.

  Lower MSE: Better! Your model's predictions are closer to the actual values.
  Higher MSE: Worse! Your model's predictions are far from the actual values.

- ***Mean Squared Error:***

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

- **R2 Score:**

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

- ***Add necessary figure(s)/Diagram(s)***

Input & Output:

About dataset and custom user input

Advertising and Sales Dataset
• Objective: This dataset explores how different types of advertising spending influence sales revenue.
• Features:

- **TV**: A numerical variable indicating the amount spent on TV advertisements.
- **Radio**: A numerical variable denoting the amount spent on radio advertisements.
- **Newspaper**: A numerical variable representing the amount spent on newspaper advertisements.

Sales: A numerical variable showing the total revenue from sales
Analyze the results: How well the model fits the data.

Energy Efficiency Dataset:

- **Objective:** This dataset is used to predict Heating Load and Cooling Load of buildings based on architectural features. It helps in analyzing energy efficiency and optimizing building designs.
- **Features:**

  - **Relative Compactness (X1):** A continuous variable representing the compactness of the building shape.
  - **Surface Area (X2):** The total surface area of the building.
  - **Wall Area (X3):** Area covered by the walls.
  - **Roof Area (X4):** Area covered by the roof.
  - **Overall Height (X5):** The height of the building.
  - **Orientation (X6):** The orientation of the building (categorical encoded as numerical).

- **Glazing Area (X7):** The percentage of window area on the exterior surface.
- **Glazing Area Distribution (X8):** The distribution pattern of glazing area on the four facades.

- **Target Variables:**

  - **Heating Load (y1):** The amount of heating required for maintaining indoor temperature.
  - **Cooling Load (y2):** The amount of cooling required for maintaining indoor temperature.

Analyze the Results: How well the model fits the data.

Heating Load Prediction (y1):

- **Ridge Regression:**
  - **R^2 Score:** 0.89
    This score indicates that 89% of the variation in Heating Load can be explained by the model. It suggests a good fit to the data.
  - **RMSE:** 2.74
    On average, the predicted Heating Load deviates from the actual value by about 2.74 units.
- **Lasso Regression:**
  - **R^2 Score:** 0.87
    Slightly lower than Ridge, indicating a good but slightly less accurate fit.
  - **RMSE:** 2.98
    The error is marginally higher than Ridge, showing a small trade-off for feature selection.

Cooling Load Prediction (y2):

- **Ridge Regression:**
  - **R^2 Score:** 0.92
    This high value suggests that 92% of the variation in Cooling Load is accounted for by the model.
  - **RMSE:** 2.12
    The predictions deviate by 2.12 units on average, showing high accuracy.
- **Lasso Regression:**
  - **R^2 Score:** 0.90
    Similar to Ridge but slightly lower, indicating good predictive power.
  - **RMSE:** 2.36
    Slightly higher error compared to Ridge but still within an acceptable range.

Challenges Encountered During the Implementation:

1. **Multicollinearity:**

   - The architectural features were highly correlated, impacting model stability. Ridge regression effectively handled this issue, while Lasso performed variable selection.
2. **Hyperparameter Tuning:**
   - Determining the optimal alpha value for both models required extensive cross-validation.
3. **Feature Selection with Lasso:**
   - Lasso set some coefficients to zero, leading to simpler models but slightly higher prediction errors compared to Ridge.
4. **Visualizing Results:**
   - Plotting actual vs. predicted values for two target variables required careful interpretation.

Conclusion:

- **Significance of Independent Variables and Their Interactions:**
  - Relative Compactness, Surface Area, and Overall Height were consistently important for both Heating and Cooling Loads, indicating that building shape significantly impacts energy efficiency.
  - Lasso regression revealed that some features like Glazing Area Distribution were less impactful, driving their coefficients to zero.
- **Nature of Relationships Based on Regression Coefficients:**
  - Positive coefficients for Relative Compactness and Overall Height indicate that as these values increase, both Heating and Cooling Loads tend to increase.
  - Negative coefficients for Orientation suggest that certain orientations are more energy-efficient.
- **Importance of Performance Metrics and Comparison:**
  - **R^2 Score** was used to measure the proportion of variance explained by the model, indicating good fits for both Ridge and Lasso.
  - **RMSE** provided an interpretation of the average prediction error, showing Ridge had slightly lower errors due to its ability to handle multicollinearity better.
  - **Ridge vs. Lasso:**
    - **Ridge Regression** produced slightly better accuracy but kept all features in the model.
    - **Lasso Regression** provided a more interpretable model by performing feature selection, albeit with a small sacrifice in prediction accuracy.