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: A method used to analyze multiple regression data that suffer from multicollinearity. It adds a penalty equal to the square of the magnitude of coefficients (L2 regularization) to the loss function, which helps in reducing model complexity.
    - Lasso Regression: Similar to Ridge, but it adds a penalty equal to the absolute value of the magnitude of coefficients (L1 regularization). This can lead to sparse models where some coefficients can become exactly zero, thus performing variable selection.
- Give mathematical formulation of Ridge and Lasso regression
    - Ridge Regression:

$$\text{Loss Function} = i=1\sum n(yi - y\hat{}i)2 + \lambda j=1\sum p\beta j2$$

- Lasso Regression:

$$\text{Loss Function} = i=1\sum n(yi - y\hat{}i)2 + \lambda j=1\sum p|\beta j|$$

Where yi is the actual value, $y\hat{}i$ is the predicted value, n is the number of observations, p is the number of predictors, βj are the coefficients, and λ is the regularization parameter.

- 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.

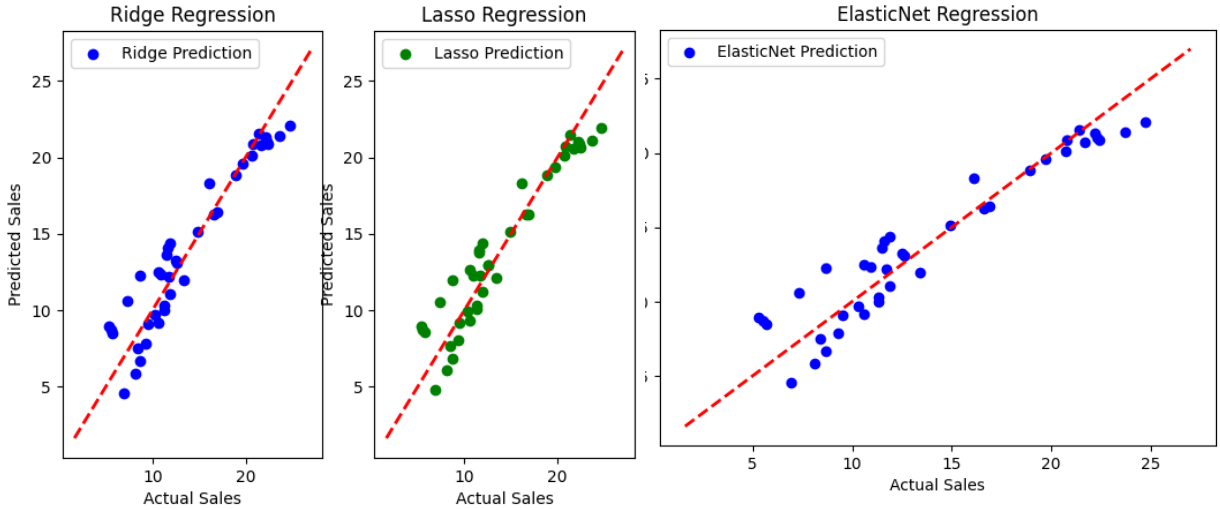
Lasso Regression:
• Performs automatic feature selection by setting irrelevant feature coefficients to zero.
• Useful for high-dimensional data where feature selection is necessary for model interpretability and efficiency.
• Enhances model generalization by reducing overfitting through sparse solutions.

- Mention applications of Ridge and Lasso regression in real-world scenarios.
  - Finance: Risk assessment models where multicollinearity exists among financial indicators.
  - Healthcare: Predicting patient outcomes based on numerous health metrics.
  - Marketing: Customer segmentation analysis with many demographic variables.

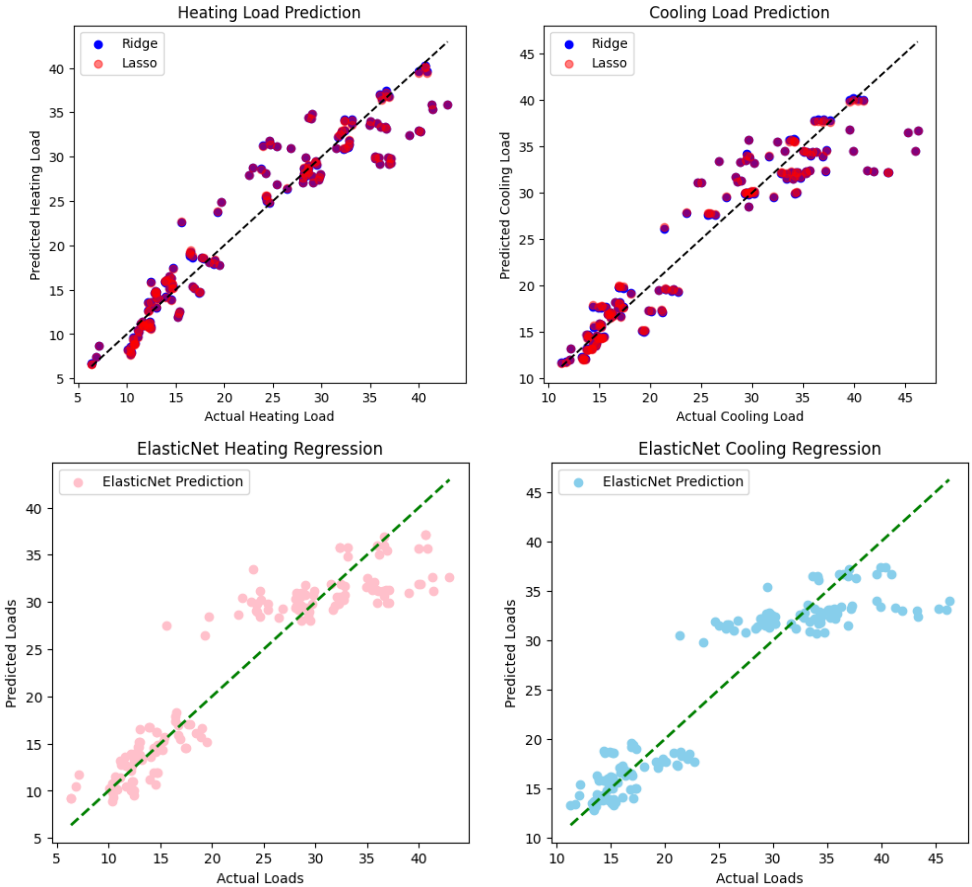
- Brief explanation of performance metrics (e.g., R^2, Mean Squared Error, Root Mean Squared Error).
  - **R^2 (Coefficient of Determination):** Indicates how well data points fit a statistical model. Ranges from 0 to 1; higher values indicate better fit.
  - **Mean Squared Error (MSE):** Measures average squared difference between observed and predicted values. Lower values indicate better model performance.
  - **Root Mean Squared Error (RMSE):** The square root of MSE; provides error in same units as dependent variable, making interpretation easier.

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

Advertising and Sales Dataset



Energy Efficiency Dataset



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

.

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:
  - o R^2 Score: 0.91
    This score indicates that 91% of the variation in Heating Load can be explained by the model. It suggests a good fit to the data.
  - o RMSE: 3.03
    On average, the predicted Heating Load deviates from the actual value by about 3.03 units.

- Lasso Regression:
  - o R^2 Score: 0.91
    This score indicates that 91% of the variation in Heating Load can be explained by the model. It suggests a good fit to the data.
  - o RMSE: 3.03
    On average, the predicted Heating Load deviates from the actual value by about 3.03 units.

Cooling Load Prediction (y2):

- Ridge Regression:
  - o R^2 Score: 0.89
    This high value suggests that 89% of the variation in Cooling Load is accounted for by the model.
  - o RMSE: 3.15
    The predictions deviate by 3.15 units on average, showing high accuracy.

- Lasso Regression:
  - o R^2 Score: 0.89
    This high value suggests that 89% of the variation in Cooling Load is accounted for by the model.
  - o RMSE: 3.15
    The predictions deviate by 3.15 units on average, showing high accuracy.

Challenges encountered during the implementation.

- **Data Scaling:** Ridge and Lasso Regression require input features to be standardized before fitting the model. Features with large ranges of values can bias results if they are not properly scaled relative to other features. Data that is not centered at the mean with a standard deviation of 1 will also need rescaling to limit the impact of large scales on the model. Unscaled features can result in the application of unintentional penalties in Lasso Regression due to differences in units.

- **Hyperparameter Tuning:** Selecting the optimal value for the regularization parameter (lambda/alpha) is critical but challenging. If the parameter is too small, the model may overfit the data. If it is too large, the model may underfit the data. Cross-validation is often used to find the value that minimizes the prediction error.

- **Multicollinearity:** While Ridge Regression is good at handling multicollinearity, Lasso Regression has limitations. Lasso may arbitrarily choose one feature to include in the model when there are correlated features. Ridge Regression can reduce the variance of coefficients by shrinking them towards zero, but it cannot eliminate the correlation among them.

- **Interpretability:** Ridge Regression can make the model more complex due to the shrinkage of coefficients, making it harder to understand the impact of each predictor on the response variable.

- **Outliers and Noise:** If the data points contain outliers or noise, this could produce inaccurate predictions due to the penalty terms.

- **Computational Cost:** Ridge and Lasso Regressions can be slow when applied to large datasets because of the computation time needed to perform regularization and hyperparameter tuning.

Conclusion:
Summarize the significance of independent variables and their interactions.

Independent variables play a crucial role in regression analysis by influencing the dependent variable, which is the outcome being measured. Their significance can be assessed through various metrics and methods:

- Contribution to Variance: Independent variables help explain variations in the dependent variable, with their contributions often quantified using the R-squared value. A higher R-squared indicates that a significant portion of the variability in the dependent variable can be explained by the independent variables, enhancing model accuracy.

- Importance of Coefficients: Each independent variable has an associated coefficient that quantifies its effect on the dependent variable while controlling for other variables. This allows analysts to identify which factors are most impactful in predicting outcomes.

- Interactions Between Variables: Understanding how independent variables interact with each other can provide deeper insights into their collective influence on the dependent variable. This is particularly important in multiple regression analysis, where multiple predictors are considered simultaneously.

- Statistical Significance: The significance of independent variables is often assessed using p-values. A low p-value indicates that there is sufficient

evidence to conclude that a non-zero correlation exists between the independent and dependent variables, making it a valuable addition to the model.

- Decision-Making Insights: By analyzing the contributions and interactions of independent variables, stakeholders can make informed decisions based on data-driven insights, guiding effective strategies and forecasting.

Discuss the nature of the relationships based on regression coefficients.

Regression coefficients reveal the nature of the relationship between independent variables and the dependent variable.

- Sign of the Coefficient:
  - Positive Coefficient: Indicates a positive or direct relationship. As the independent variable (X) increases, the dependent variable (Y) also tends to increase.
  - Negative Coefficient: Indicates a negative or inverse relationship. As the independent variable (X) increases, the dependent variable (Y) tends to decrease.

- Magnitude of the Coefficient: The coefficient's absolute value indicates the strength of the relationship. A larger coefficient suggests a more significant change in the dependent variable for each unit change in the independent variable.

- Holding Other Variables Constant: Regression coefficients describe the change in the dependent variable associated with a one-unit change in the independent variable, assuming all other independent variables in the model are held constant.

- Regression Equation: In a linear regression equation ($Y = aX + b$), 'a' represents the regression coefficient, showing how much Y changes for each unit change in X.

Highlight the importance of performance metrics and compare it for both Ridge and Lasso regression

Importance of performance metrics

Performance metrics are crucial for businesses because they provide a quantifiable way to assess the effectiveness of business processes and strategies.

- Measuring Progress and Growth: They help in assessing the state of the business and measuring progress toward objectives.

- Identifying Operational Problems: Performance metrics enable businesses to identify areas that need improvement.

- Informing Future Decisions: They provide data-driven insights that guide effective strategies and forecasting.
- Promoting Accountability: By monitoring performance indicators, businesses can hold people accountable and incentivize success.
- Evaluating Performance: Performance metrics create a structure for evaluating employee achievements.
- Enabling Comparison: They allow for the comparison of data to pre-defined objectives or targets.

Comparison of Performance Metrics for Ridge and Lasso Regression

Metric	Ridge Regression	Lasso Regression
Mean Squared Error (MSE)	Measures model accuracy by penalizing large errors. Ridge and Lasso both minimize MSE, but Ridge is better at handling correlated features.	Similar to Ridge, but Lasso may lead to sparse models by eliminating some coefficients.
R^2 (Coefficient of Determination)	Indicates how well the model explains variance in the dependent variable. Ridge generally retains more features and distributes weights more evenly.	Lasso may lead to a lower R^2 if many coefficients are shrunk to zero.
Mean Absolute Error (MAE)	Measures average absolute error, less sensitive to outliers than MSE.	Performs similarly but can be more interpretable due to feature selection.
Regularization Effect	Reduces overfitting by shrinking coefficients but retains all variables.	Can set some coefficients to zero, effectively performing feature selection.
Computational Complexity	Slightly faster as it solves a simple closed-form equation.	May be slower due to the need for feature selection.

Ridge and Lasso Regression

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error
```

In [2]:
```
df = pd.read_csv("Assignment 2 Advertising.csv", index_col=0)
df.head()
```

Out[2]:

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

In [3]:
```
df.shape
```

Out[3]:
```
(200, 4)
```

In [4]:
```
X = df[['TV', 'Radio', 'Newspaper']]
y = df['Sales']
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

Hyperparameter:

alpha: Controls the strength of regularisation

- Higher alpha - High regularisaton
- Lower alpha - Low regularisaton

In [6]:
```
alpha_values = np.logspace(-3, 3, 10)
# 10 values from 0.001 to 1000 (10^-3 to 10^3)
print(alpha_values)
```

```
[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]
```

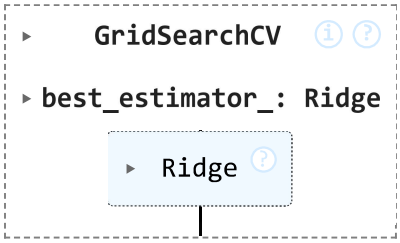
In [7]:
```
alpha_values = {'alpha': np.logspace(-3, 3, 10)}
```

Ridge Regression

In [8]:
```
ridge = Ridge()
```

In [9]:
```
ridge_cv = GridSearchCV(ridge, alpha_values, cv=5, scoring='neg_mean_squared_err
ridge_cv.fit(X_train, y_train)
```

Out[9]:



In [10]:
```
best_ridge = ridge_cv.best_estimator_
y_pred_ridge = best_ridge.predict(X_test)
```

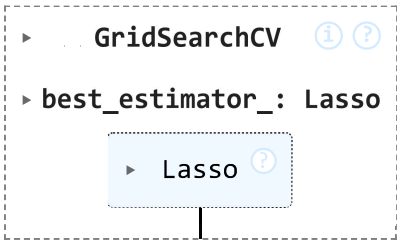
In [11]:
```
rmse_ridge = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
print(f"Best Ridge Alpha: {ridge_cv.best_params_['alpha']}")
print(f"Ridge Regression RMSE: {rmse_ridge:.2f}")
```

```
Best Ridge Alpha: 0.001
Ridge Regression RMSE: 1.78
```

Lasso Regression

In [12]:
```
lasso = Lasso()
lasso_cv = GridSearchCV(lasso, alpha_values, cv=5, scoring='neg_mean_squared_err
lasso_cv.fit(X_train, y_train)
```

Out[12]:



In [13]:
```
best_lasso = lasso_cv.best_estimator_
y_pred_lasso = best_lasso.predict(X_test)
```

In [14]:
```
rmse_lasso = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
print(f"Best Lasso Alpha: {lasso_cv.best_params_['alpha']}")
print(f"Lasso Regression RMSE: {rmse_lasso:.2f}")
```

```
Best Lasso Alpha: 2.154434690031882
Lasso Regression RMSE: 1.77
```

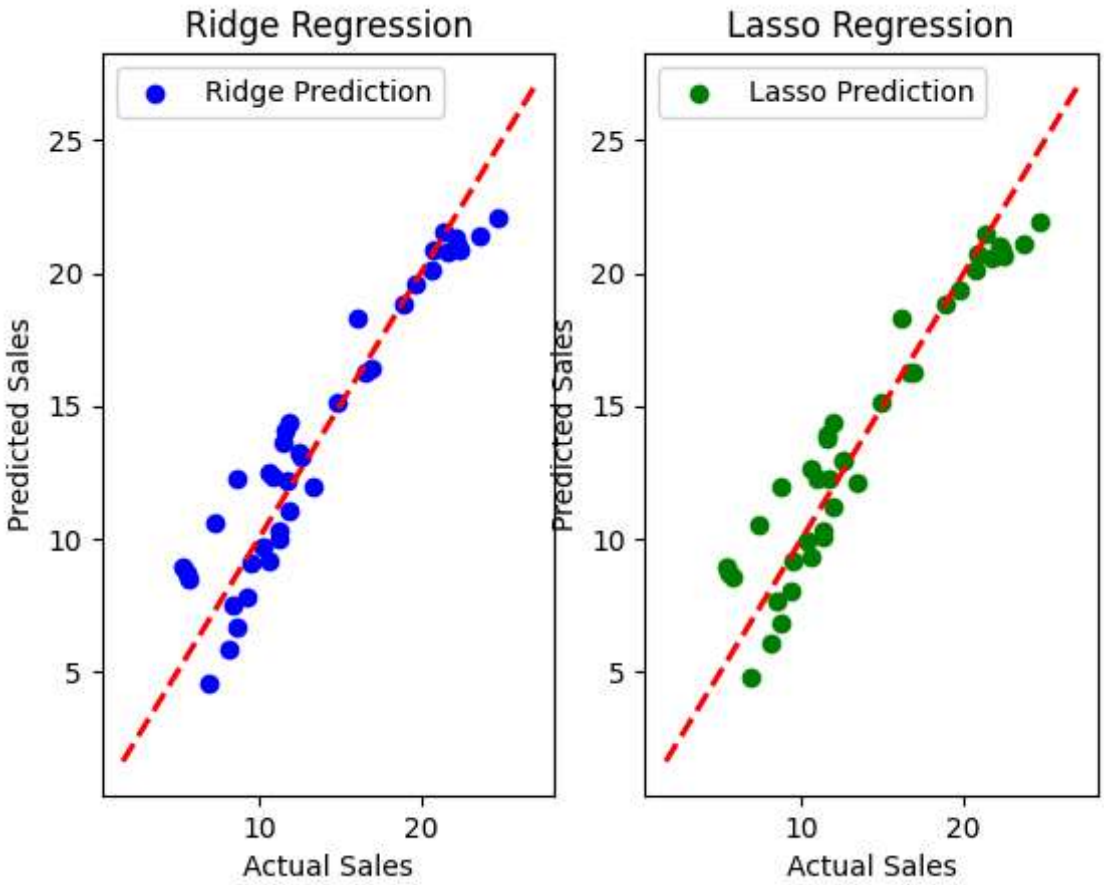
In [15]:
```
plt.figure(figsize=(10, 5))
```

Out[15]:
```
<Figure size 1000x500 with 0 Axes>
```

```
<Figure size 1000x500 with 0 Axes>
```

In [16]:
```
# Ridge Plot
plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred_ridge, color='blue', label="Ridge Prediction")
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 Plot
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_lasso, color='green', label="Lasso Prediction")
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()
```



In [17]:
```
# Ridge Coeffient
for feature_name, coef in zip(X.columns, best_ridge.coef_):
    print(f"Featues: {feature_name}, Coeffient: {coef}")
```

```
Featues: TV, Coeffient: 0.044729517481338306
Featues: Radio, Coeffient: 0.18919504786574776
Featues: Newspaper, Coeffient: 0.0027611160976652995
```

In [18]:
```
# Lasso Coeffient
for feature_name, coef in zip(X.columns, best_lasso.coef_):
    print(f"Featues: {feature_name}, Coeffient: {coef}")
```

```
Featues: TV, Coeffient: 0.044516937884577404
Featues: Radio, Coeffient: 0.18084149925032336
Featues: Newspaper, Coeffient: 0.0
```

In [19]:
```
from sklearn.linear_model import ElasticNet

elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)  # 50% L1, 50% L2
elastic_net.fit(X_train, y_train)
```

Out[19]:

```
ElasticNet
ElasticNet(alpha=0.1)
```

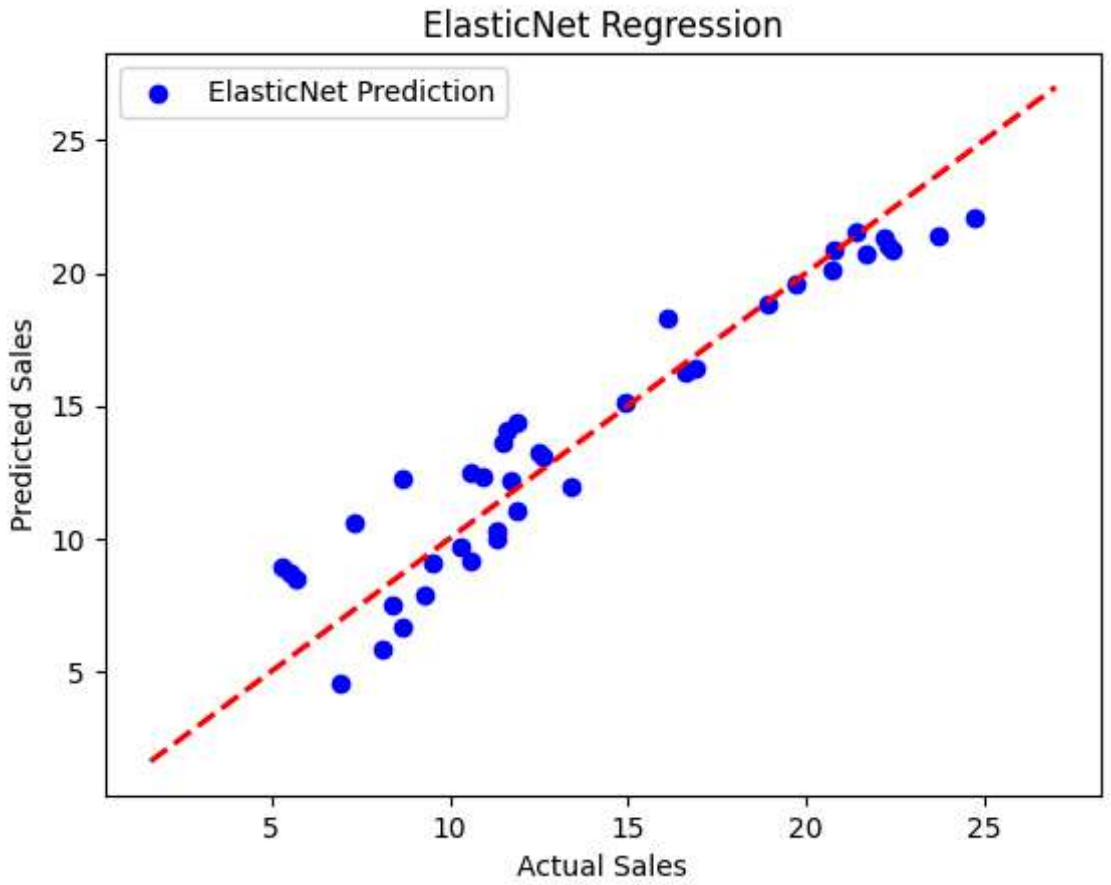
In [20]:
```
# Predictions
y_pred_elastic = elastic_net.predict(X_test)
```

In [21]:
```
# Evaluation
rmse = np.sqrt(mean_squared_error(y_test, y_pred_elastic))
print(f"Best ElasticNet Alpha: {elastic_net.alpha}")
print("ElasticNet Regression RMSE:", rmse)
```

```
Best ElasticNet Alpha: 0.1
ElasticNet Regression RMSE: 1.7810894764961034
```

In [22]:
```
plt.scatter(y_test, y_pred_elastic, color='blue', label="ElasticNet Prediction")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)
plt.xlabel('Actual Sales')
plt.ylabel("Predicted Sales")
plt.title("ElasticNet Regression")
plt.legend()
```

Out[22]:
```
<matplotlib.legend.Legend at 0x1a97fe98cd0>
```



In [23]:
```
# ElasticNet Coeffient
for feature_name, coef in zip(X.columns, elastic_net.coef_):
    print(f"Featues: {feature_name}, Coeffient: {coef}")
```

```
Featues: TV, Coeffient: 0.04472464318175472
Featues: Radio, Coeffient: 0.1889525462100257
Featues: Newspaper, Coeffient: 0.0027080267082268666
```

Energy Efficiency

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
df = pd.read_excel("Practice dataset.xlsx")
df.head()
```

Out[ ]:

	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

Exploratory Data Analysis

In [3]:
```
df.shape
```

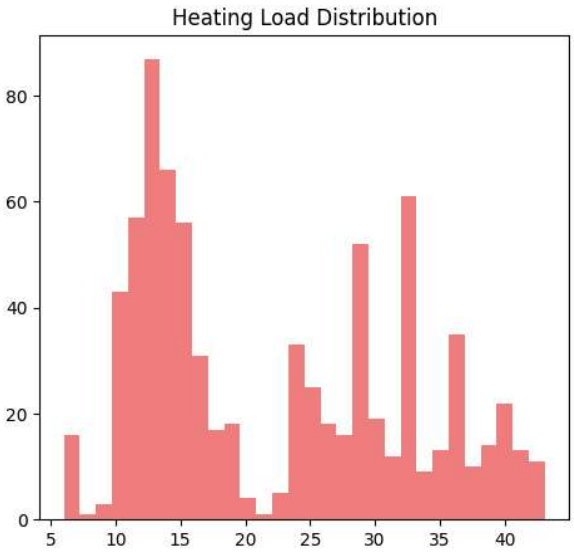
Out[3]:
```
(768, 10)
```

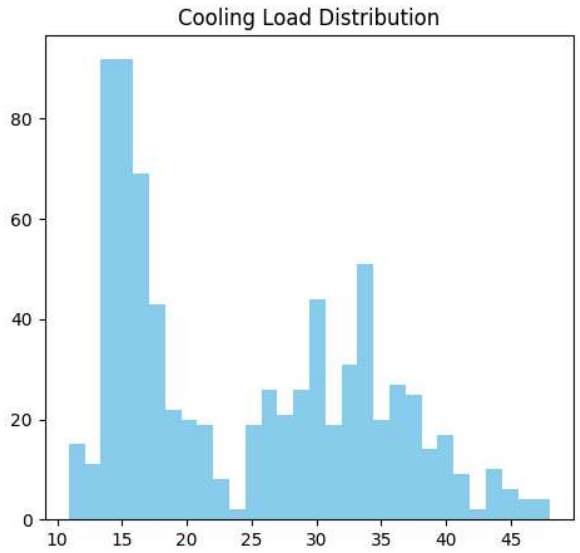
In [4]:
```
# Check for missing values
print(df.isnull().sum())
```

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

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

```
plt.title('Cooling Load Distribution')
plt.show()
```





In [6]:
```
df.describe(include='all')
```

Out[6]:

	X1	X2	X3	X4	X5	X6	X7
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000

Column Renaming

In [7]:
```
df = df.rename(columns={'X1': 'Relative Compactness', 'X2':'Surface Area','X3':'Wal
df.head()
```

Out[7]:

	Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heatin Loa
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.5
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.5
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.5
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.5
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.8

In [8]:
```
# Feature variables
X = df[['Relative Compactness', 'Surface Area', 'Wall Area', 'Roof Area',
    'Overall Height', 'Orientation', 'Glazing Area', 'Glazing Area Distribution']]
# Target variables
y1 = df['Heating Load']
y2 = df['Cooling Load']
```

In [9]:
```
# Split data for Heating Load
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y1, test_size=0.2, rando
# Split data for Cooling Load
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y2, test_size=0.2, rando
```

Ridge Regression

In [10]:
```
ridge = Ridge()
alpha_values = {'alpha': np.logspace(-3, 3, 10)}
```

In [11]:
```
# Ridge Regression for Heating Load
ridge_cv1 = GridSearchCV(ridge, alpha_values, cv=5, scoring='neg_mean_squared_error
ridge_cv1.fit(X_train1, y_train1)

# Predictions and Evaluation for Heating Load
best_ridge1 = ridge_cv1.best_estimator_
y_pred_ridge1 = best_ridge1.predict(X_test1)
rmse_ridge1 = np.sqrt(mean_squared_error(y_test1, y_pred_ridge1))
print(f"Heating Load - Ridge Regression")
print(f"Best alpha value: {ridge_cv1.best_params_['alpha']}")
print(f"RMSE: {rmse_ridge1:.2f}")
print(f"R2 Score: {r2_score(y_test1, y_pred_ridge1):.2f}")

# Ridge Regression for Cooling Load
ridge_cv2 = GridSearchCV(ridge, alpha_values, cv=5, scoring='neg_mean_squared_error
ridge_cv2.fit(X_train2, y_train2)

# Predictions and Evaluation for Cooling Load
best_ridge2 = ridge_cv2.best_estimator_
y_pred_ridge2 = best_ridge2.predict(X_test2)
rmse_ridge2 = np.sqrt(mean_squared_error(y_test2, y_pred_ridge2))
print(f"\nCooling Load - Ridge Regression")
```

```
print(f"Best alpha value: {ridge_cv2.best_params_['alpha']}")
print(f"RMSE: {rmse_ridge2:.2f}")
print(f"R2 Score: {r2_score(y_test2, y_pred_ridge2):.2f}")
```

```
Heating Load - Ridge Regression
Best alpha value: 0.001
RMSE: 3.03
R2 Score: 0.91

Cooling Load - Ridge Regression
Best alpha value: 0.001
RMSE: 3.15
R2 Score: 0.89
```

Lasso Regression

In [12]:
```
lasso = Lasso()
```

In [13]:
```
# Lasso Regression for Heating Load
lasso_cv1 = GridSearchCV(lasso, alpha_values, cv=5, scoring='neg_mean_squared_error
lasso_cv1.fit(X_train1, y_train1)

# Predictions and Evaluation for Heating Load
best_lasso1 = lasso_cv1.best_estimator_
y_pred_lasso1 = best_lasso1.predict(X_test1)
rmse_lasso1 = np.sqrt(mean_squared_error(y_test1, y_pred_lasso1))
print(f"\nHeating Load - Lasso Regression")
print(f"Best alpha value: {lasso_cv1.best_params_['alpha']}")
print(f"RMSE: {rmse_lasso1:.2f}")
print(f"R2 Score: {r2_score(y_test1, y_pred_lasso1):.2f}")

# Lasso Regression for Cooling Load
lasso_cv2 = GridSearchCV(lasso, alpha_values, cv=5, scoring='neg_mean_squared_error
lasso_cv2.fit(X_train2, y_train2)

# Predictions and Evaluation for Cooling Load
best_lasso2 = lasso_cv2.best_estimator_
y_pred_lasso2 = best_lasso2.predict(X_test2)
rmse_lasso2 = np.sqrt(mean_squared_error(y_test2, y_pred_lasso2))
print(f"\nCooling Load - Lasso Regression")
print(f"Best alpha value: {lasso_cv2.best_params_['alpha']}")
print(f"RMSE: {rmse_lasso2:.2f}")
print(f"R2 Score: {r2_score(y_test2, y_pred_lasso2):.2f}")
```

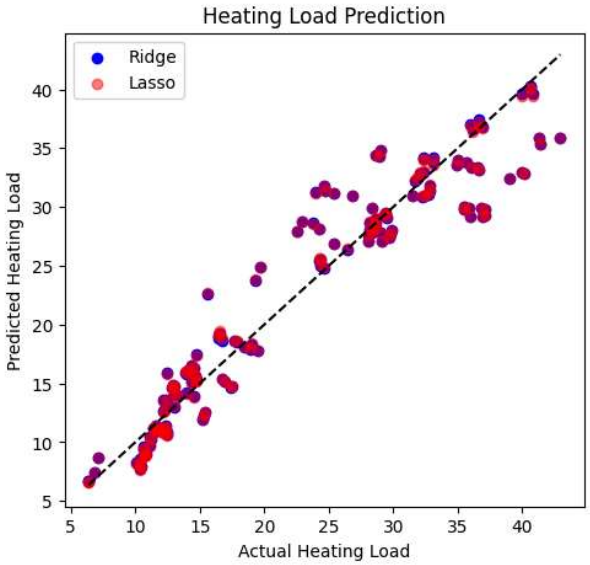
```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 1.228e+02, tolerance: 4.917e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 9.631e+01, tolerance: 4.966e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 1.167e+02, tolerance: 4.813e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 3.230e+01, tolerance: 5.017e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 2.731e+02, tolerance: 5.069e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 1.491e+02, tolerance: 6.197e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 1.922e+02, tolerance: 4.391e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 3.735e+02, tolerance: 4.368e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 2.637e+02, tolerance: 4.313e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 1.806e+02, tolerance: 4.488e+00
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 4.213e+02, tolerance: 4.459e+00
  model = cd_fast.enet_coordinate_descent(
```

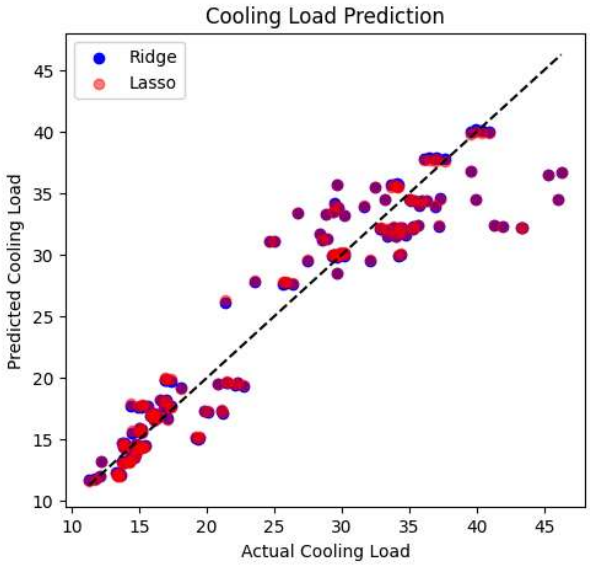
```
Heating Load - Lasso Regression
Best alpha value: 0.001
RMSE: 3.03
R2 Score: 0.91

Cooling Load - Lasso Regression
Best alpha value: 0.001
RMSE: 3.15
R2 Score: 0.89
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:
695: ConvergenceWarning: Objective did not converge. You might want to increase the
number of iterations, check the scale of the features or consider increasing regular
isation. Duality gap: 3.472e+02, tolerance: 5.505e+00
  model = cd_fast.enet_coordinate_descent(
```

Visualisation of Ridge and Lasso Regression

In [14]:
```
# Ridge vs Lasso Comparison for Heating 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', alpha=0.5)
plt.plot([min(y_test1), max(y_test1)], [min(y_test1), max(y_test1)], color='black',
plt.xlabel('Actual Heating Load')
plt.ylabel('Predicted Heating Load')
plt.title('Heating Load Prediction')
plt.legend()

# Ridge vs Lasso Comparison for Cooling Load
plt.subplot(1, 2, 2)
plt.scatter(y_test2, y_pred_ridge2, color='blue', label='Ridge')
plt.scatter(y_test2, y_pred_lasso2, color='red', label='Lasso', alpha=0.5)
plt.plot([min(y_test2), max(y_test2)], [min(y_test2), max(y_test2)], color='black',
plt.xlabel('Actual Cooling Load')
plt.ylabel('Predicted Cooling Load')
plt.title('Cooling Load Prediction')
plt.legend()
plt.show()
```





In [15]:
```
# Feature Coefficients for Heating Load
print("\nHeating Load - Lasso Coefficients")
for feature_name, coef in zip(X.columns, best_lasso1.coef_):
    print(f"Feature: {feature_name}, Coefficient: {coef}")

# Feature Coefficients for Cooling Load
print("\nCooling Load - Lasso Coefficients")
for feature_name, coef in zip(X.columns, best_lasso2.coef_):
    print(f"Feature: {feature_name}, Coefficient: {coef}")
```

```
Heating Load - Lasso Coefficients
Feature: Relative Compactness, Coefficient: -52.605911137632155
Feature: Surface Area, Coefficient: -0.06480398823045869
Feature: Wall Area, Coefficient: 0.05347820741498843
Feature: Roof Area, Coefficient: -0.010348894348998935
Feature: Overall Height, Coefficient: 4.311757167938368
Feature: Orientation, Coefficient: -0.03055485518451142
Feature: Glazing Area, Coefficient: 20.107925702683392
Feature: Glazing Area Distribution, Coefficient: 0.21248453753709526

Cooling Load - Lasso Coefficients
Feature: Relative Compactness, Coefficient: -61.8103952304079
Feature: Surface Area, Coefficient: -0.07429360331199952
Feature: Wall Area, Coefficient: 0.041731385349370355
Feature: Roof Area, Coefficient: -0.00807569510822555
Feature: Overall Height, Coefficient: 4.2342379842548175
Feature: Orientation, Coefficient: 0.055461065531980255
Feature: Glazing Area, Coefficient: 14.752726674741707
Feature: Glazing Area Distribution, Coefficient: 0.035127239570728046
```

ElasticNet Regression

In [16]:
```
from sklearn.linear_model import ElasticNet

elastic_net_y1 = ElasticNet(alpha=0.1, l1_ratio=0.5)  # 50% L1, 50% L2
elastic_net_y1.fit(X_train1, y_train1)
```

```
elastic_net_y2 = ElasticNet(alpha=0.1, l1_ratio=0.5)  # 50% L1, 50% L2
elastic_net_y2.fit(X_train2, y_train2)
```

Out[16]:

ElasticNet

ElasticNet(alpha=0.1)

In [17]:
```
# Predictions
y_pred_elastic_y1 = elastic_net_y1.predict(X_test1)

y_pred_elastic_y2 = elastic_net_y2.predict(X_test2)
```

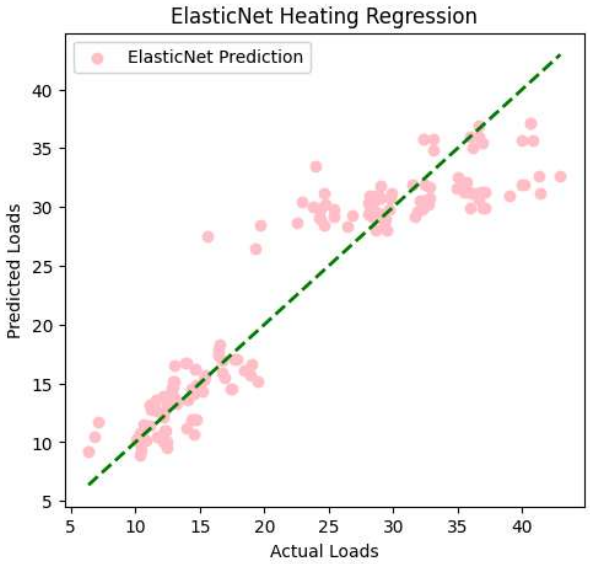
In [18]:
```
# Evaluation for Heating Load
rmse1 = np.sqrt(mean_squared_error(y_test1, y_pred_elastic_y1))
print(f"\nHeating Load - ElasticNet Regression")
print(f"RMSE: {rmse1:.2f}")
print(f"R2 Score: {r2_score(y_test1, y_pred_elastic_y1):.2f}")

# Evaluation for Cooling Load
rmse2 = np.sqrt(mean_squared_error(y_test2, y_pred_elastic_y2))
print(f"\nCooling Load - ElasticNet Regression")
print(f"RMSE: {rmse2:.2f}")
print(f"R2 Score: {r2_score(y_test2, y_pred_elastic_y2):.2f}")
```

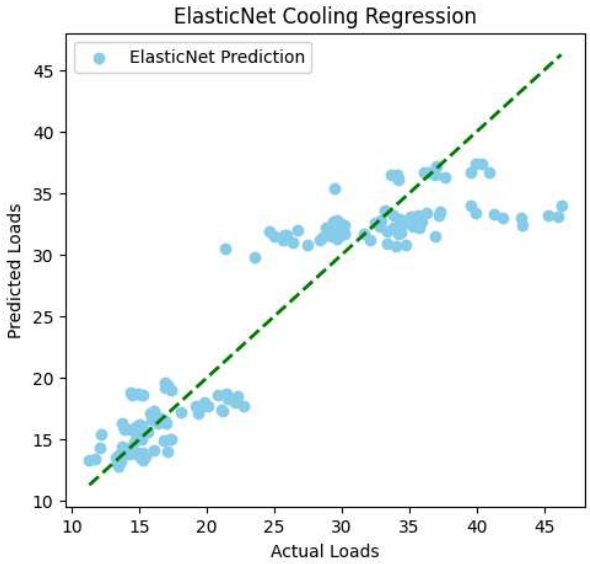
```
Heating Load - ElasticNet Regression
RMSE: 3.66
R2 Score: 0.87

Cooling Load - ElasticNet Regression
RMSE: 3.62
R2 Score: 0.86
```

ElasticNet Visualisation

In [19]:
```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(y_test1, y_pred_elastic_y1, color='pink', label="ElasticNet Prediction"
plt.plot([y_test1.min(), y_test1.max()], [y_test1.min(), y_test1.max()], 'g--', lw=
plt.xlabel('Actual Loads')
plt.ylabel("Predicted Loads")
plt.title("ElasticNet Heating Regression")
plt.legend()

plt.subplot(1, 2, 2)
plt.scatter(y_test2, y_pred_elastic_y2, color='skyblue', label="ElasticNet Predicti
plt.plot([y_test2.min(), y_test2.max()], [y_test2.min(), y_test2.max()], 'g--', lw=
plt.xlabel('Actual Loads')
plt.ylabel("Predicted Loads")
plt.title("ElasticNet Cooling Regression")
plt.legend()
```

Out[19]: <matplotlib.legend.Legend at 0x79784bba9e50>


ElasticNet Heating Regression
ElasticNet Prediction
Predicted Loads
Actual Loads
5
10
15
20
25
30
35
40


ElasticNet Cooling Regression
ElasticNet Prediction
Predicted Loads
Actual Loads
10
15
20
25
30
35
40
45

In [19]: