```
from google.colab import files

uploaded = files.upload()
```

Choose Files no files selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

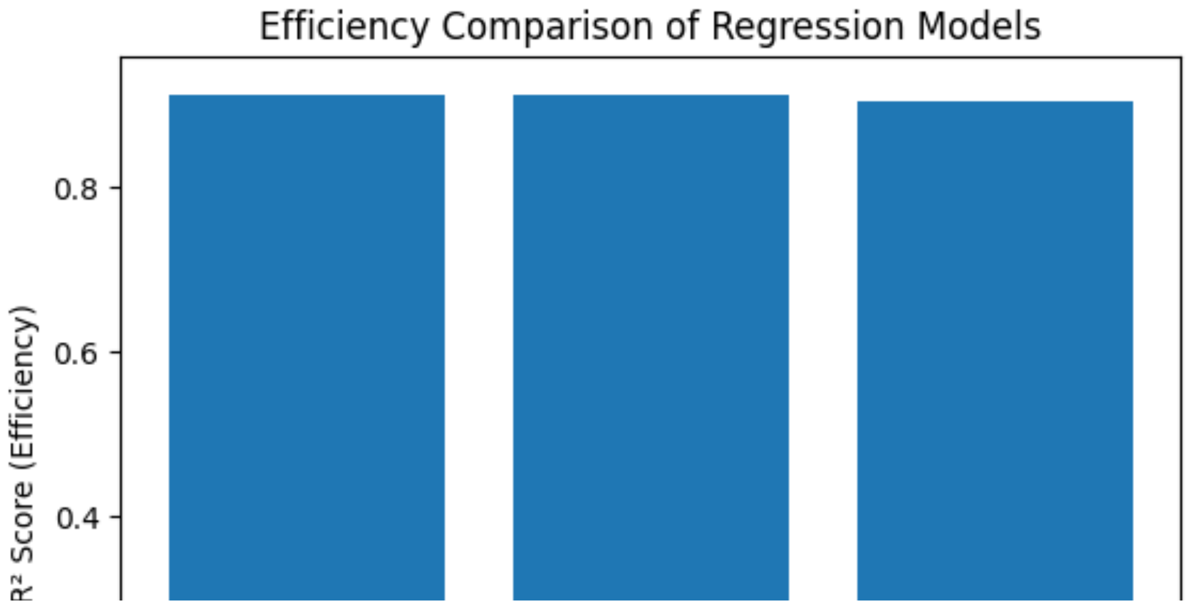
```
Saving energy+efficiency.zip to energy+efficiency.zip
```

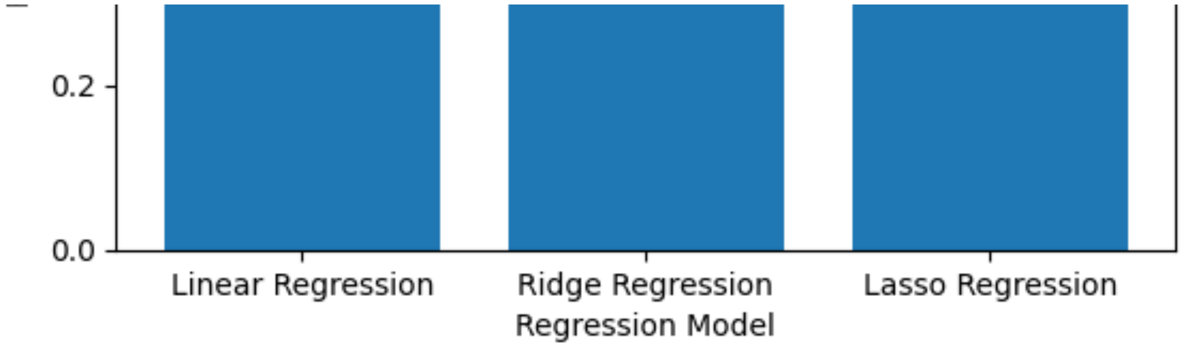
```
# ================================
# 1. Import Required Libraries
# ================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import zipfile # Added for unzipping the dataset

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score



# ================================
# 2. Load Dataset
# ================================
# Unzip the uploaded dataset
with zipfile.ZipFile('energy+efficiency.zip', 'r') as zip_ref:
    zip_ref.extractall('.')

df = pd.read_excel('ENB2012_data.xlsx')

df.columns = [
    'Relative_Compactness', 'Surface_Area', 'Wall_Area', 'Roof_Area'
    'Overall_Height', 'Orientation', 'Glazing_Area',
    'Glazing_Area_Distribution', 'Heating_Load', 'Cooling_Load'
]

print("Dataset Loaded Successfully")
print(df.head())



# ================================
# 3. Features & Target
# ================================
X = df.drop(['Heating_Load', 'Cooling_Load'], axis=1)
```

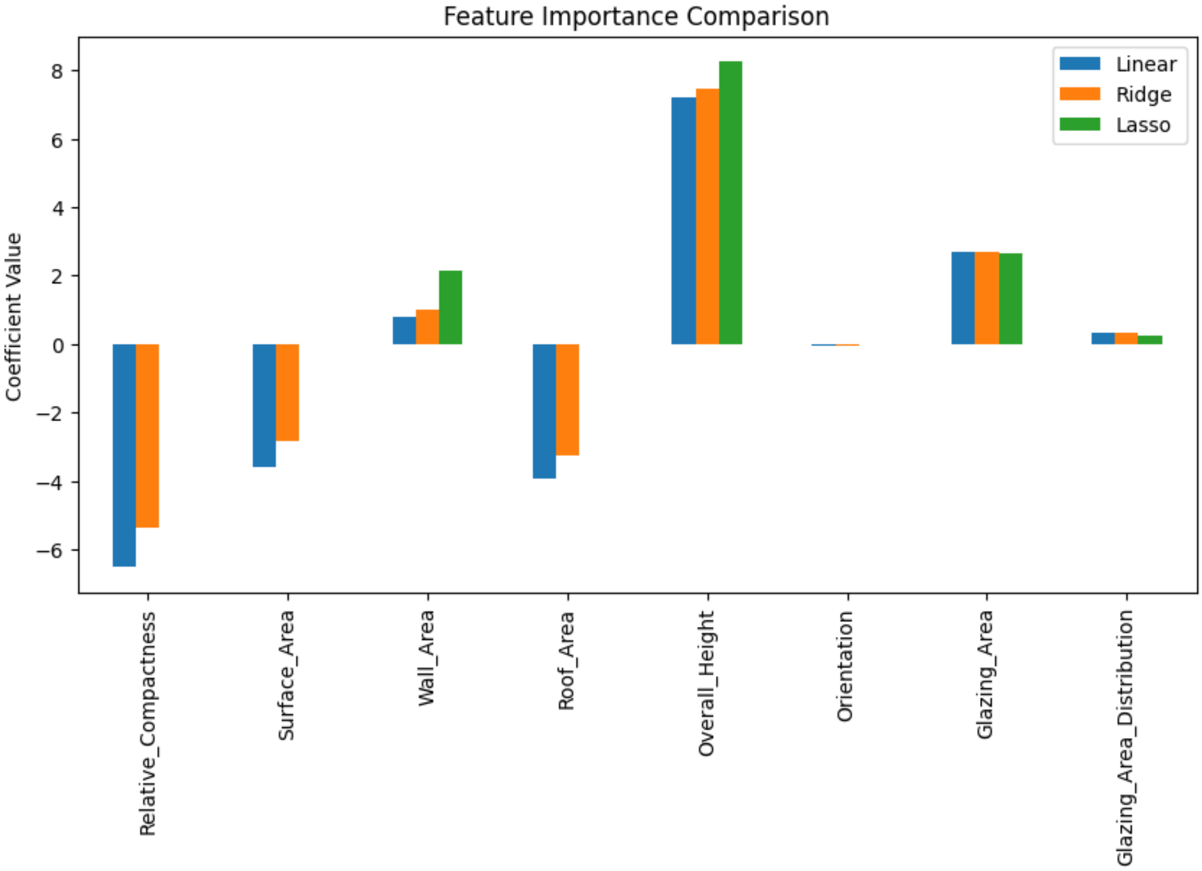
```
y = df['Heating_Load']


# ================================
# 4. Train-Test Split
# ================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)


# ================================
# 5. Feature Scaling
# ================================
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


# ================================
# 6. Train Models
# ================================
linear = LinearRegression()
ridge = Ridge(alpha=1.0)
lasso = Lasso(alpha=0.1)

linear.fit(X_train_scaled, y_train)
ridge.fit(X_train_scaled, y_train)
lasso.fit(X_train_scaled, y_train)


# ================================
# 7. Predictions
# ================================
y_pred_linear = linear.predict(X_test_scaled)
y_pred_ridge = ridge.predict(X_test_scaled)
y_pred_lasso = lasso.predict(X_test_scaled)


# ================================
# 8. Performance Evaluation
# ================================
results = pd.DataFrame({
    "Model": ["Linear Regression", "Ridge Regression", "Lasso Regres
    "MSE": [
        mean_squared_error(y_test, y_pred_linear),
        mean_squared_error(y_test, y_pred_ridge),
        mean_squared_error(y_test, y_pred_lasso)
    ],
    "R2 Score": [
        r2_score(y_test, y_pred_linear),
        r2_score(y_test, y_pred_ridge),
        r2_score(y_test, y_pred_lasso)
    ]
})

print("\nModel Efficiency Comparison:")
print(results)


# ===============================
# 9. Lasso Feature Selection
# ===============================
lasso_coeff = pd.Series(lasso.coef_, index=X.columns)

kept_features = lasso_coeff[lasso_coeff != 0].index.tolist()
removed_features = lasso_coeff[lasso_coeff == 0].index.tolist()

print("\n✅ Features KEPT by Lasso:")
print(kept_features)

print("\n❌ Features REMOVED by Lasso:")
print(removed_features)


# ===============================
# 10. Efficiency Graph (R² Score)
# ===============================
plt.figure()
plt.bar(results["Model"], results["R2 Score"])
plt.xlabel("Regression Model")
plt.ylabel("R² Score (Efficiency)")
plt.title("Efficiency Comparison of Regression Models")
plt.show()


# ===============================
# 11. Coefficient Comparison
# ===============================
coef_df = pd.DataFrame({
    "Linear": linear.coef_,
    "Ridge": ridge.coef_,
    "Lasso": lasso.coef_
}, index=X.columns)

coef_df.plot(kind='bar', figsize=(10,5))
```

```
plt.ylabel("Coefficient Value")
plt.title("Feature Importance Comparison")
plt.show()
```

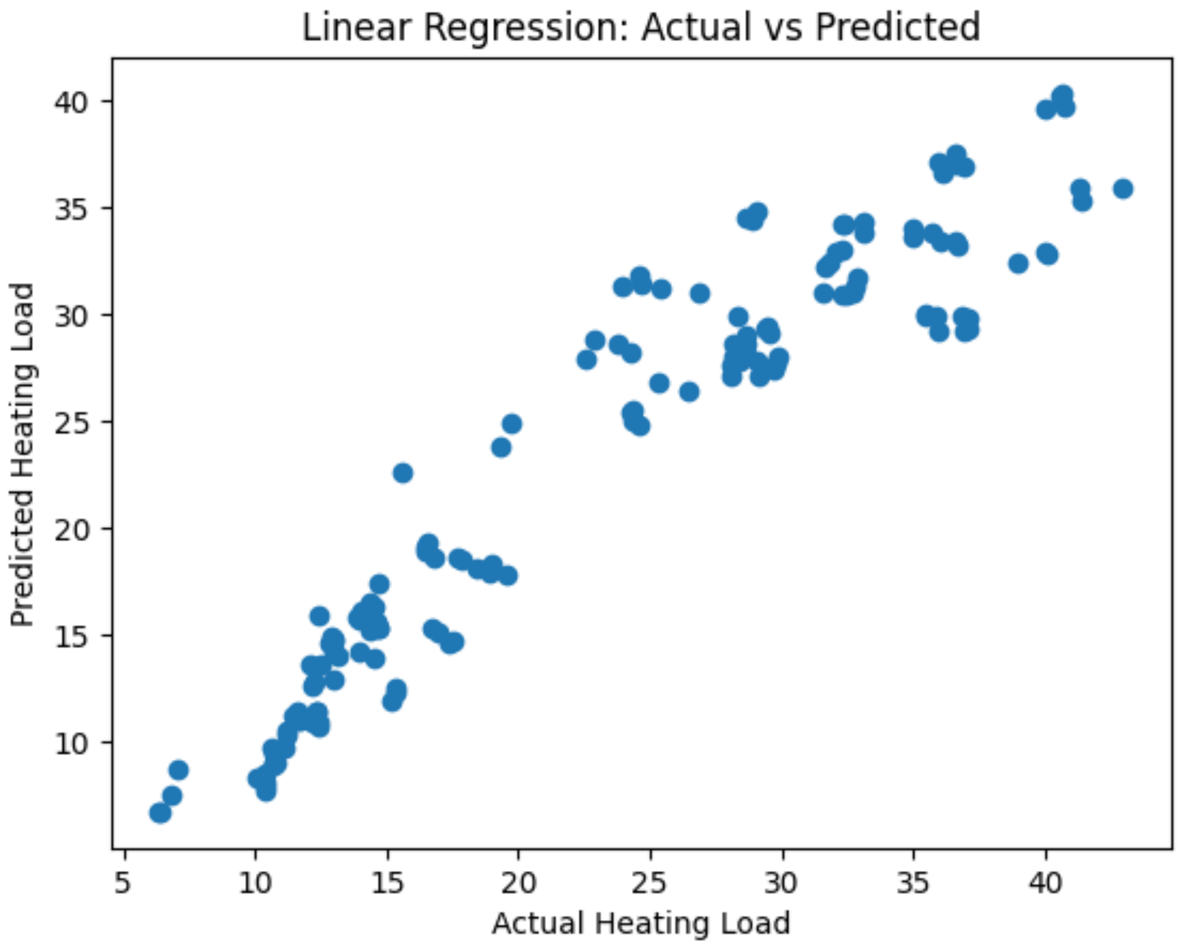
```
Dataset Loaded Successfully
   Relative_Compactness  Surface_Area  Wall_Area  Roof_Area  Overall
0                  0.98         514.5      294.0     110.25
1                  0.98         514.5      294.0     110.25
2                  0.98         514.5      294.0     110.25
3                  0.98         514.5      294.0     110.25
4                  0.90         563.5      318.5     122.50

   Orientation  Glazing_Area  Glazing_Area_Distribution  Heating_Loa
0            2           0.0                          0         15.5
1            3           0.0                          0         15.5
2            4           0.0                          0         15.5
3            5           0.0                          0         15.5
4            2           0.0                          0         20.8

   Cooling_Load
0         21.33
1         21.33
2         21.33
3         21.33
4         28.28

Model Efficiency Comparison:
               Model       MSE  R2 Score
0  Linear Regression  9.153208  0.912185
1   Ridge Regression  9.213843  0.911603
2   Lasso Regression  9.938754  0.904648

✅ Features KEPT by Lasso:
['Relative_Compactness', 'Wall_Area', 'Overall_Height', 'Glazing_Are

❌ Features REMOVED by Lasso:
['Surface_Area', 'Roof_Area', 'Orientation']
```



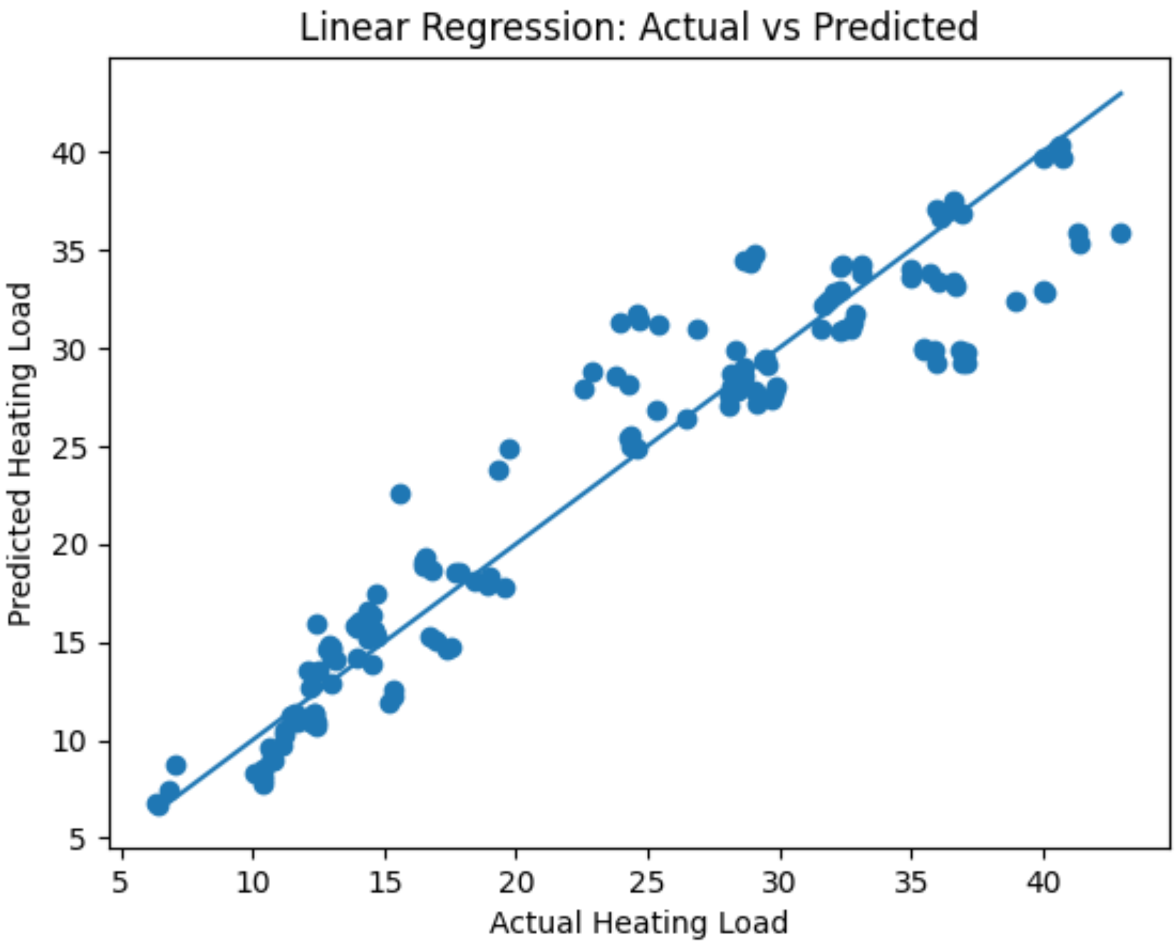

0.2
0.0
Linear Regression
Ridge Regression
Lasso Regression
Regression Model


Feature Importance Comparison
Coefficient Value
8
6
4
2
0
−2
−4
−6
Linear
Ridge
Lasso
Relative_Compactness
Surface_Area
Wall_Area
Roof_Area
Overall_Height
Orientation
Glazing_Area
Glazing_Area_Distribution

```
plt.figure()
plt.scatter(y_test, y_pred_linear)
plt.xlabel("Actual Heating Load")
plt.ylabel("Predicted Heating Load")
plt.title("Linear Regression: Actual vs Predicted")
plt.show()
```



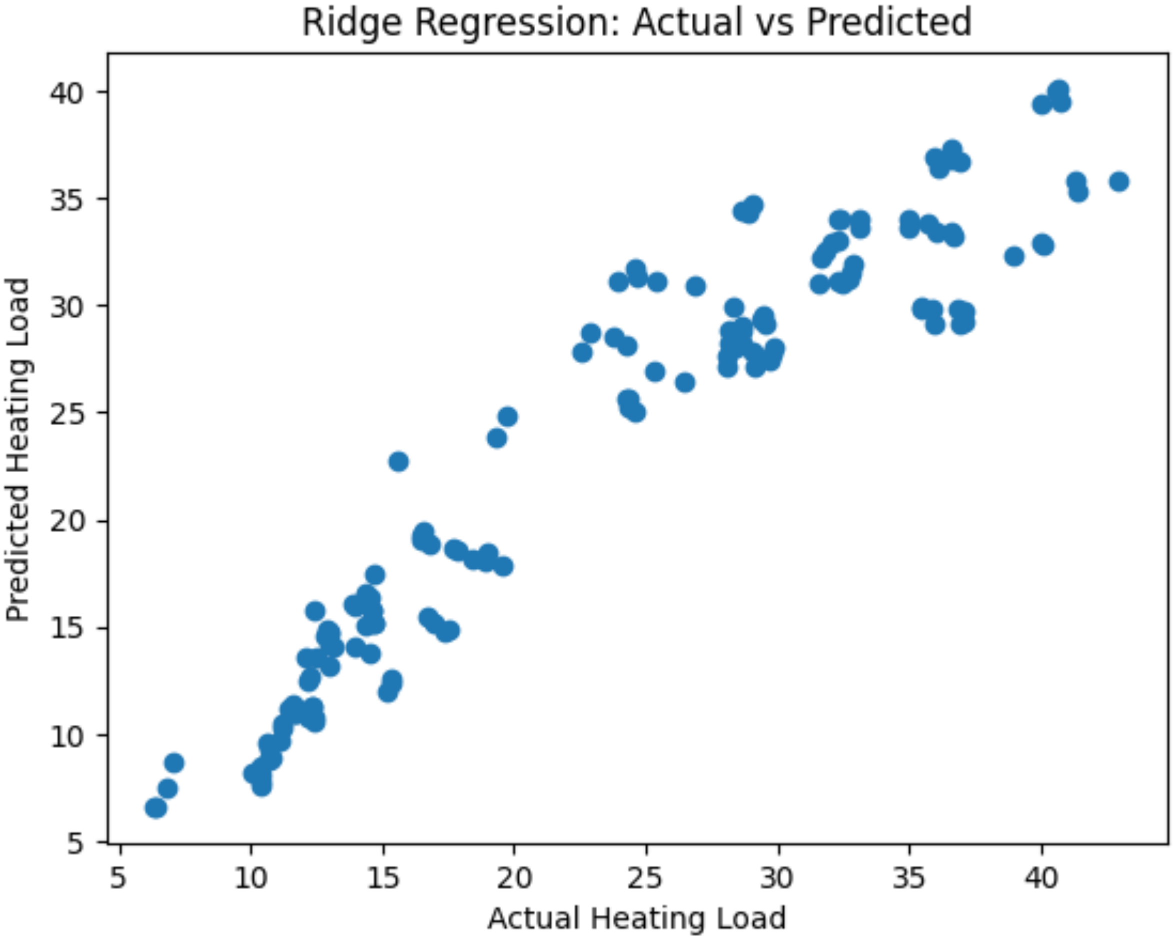
```
plt.figure()
plt.scatter(y_test, y_pred_linear)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()])
plt.xlabel("Actual Heating Load")
plt.ylabel("Predicted Heating Load")
plt.title("Linear Regression: Actual vs Predicted")
plt.show()
```



```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
plt.figure()
plt.scatter(y_test, y_pred_ridge)
plt.xlabel("Actual Heating Load")
plt.ylabel("Predicted Heating Load")
plt.title("Ridge Regression: Actual vs Predicted")
plt.show()
```


Ridge Regression: Actual vs Predicted
Predicted Heating Load
Actual Heating Load