جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MAHRAZ FES

Master Advanced Machine Learning and Multimedia Intelligence

Prediction of Building Energy Efficiency Using ELM and BP Models

Collaborators:

Mohammed El Idrissi Laoukili

Mohamed Khalil Azroul

Supervisor:

Prof. El bourkadi Dounia

Academic Year: 2024/2025

Introduction

This project aims to develop a machine learning model to predict the heating and cooling loads of buildings based on various construction parameters. It uses the *ENB2012_data.xlsx* dataset and applies regression models such as Extreme Learning Machine (ELM) and Backpropagation (BP).

I. Objectives

- Analyze the characteristics of buildings that impact energy efficiency.
- Develop and train regression models (ELM and BP) to predict heating and cooling loads.
- Evaluate and compare the models based on performance metrics.

II. Dataset Description

The dataset contains 768 samples and 8 features, with two target variables:

- Features (X1 - X8):
  - ✓ X1: Relative Compactness (real value)
  - ✓ X2: Surface Area (real value)
  - ✓ X3: Wall Area (real value)
  - ✓ X4: Roof Area (real value)
  - ✓ X5: Overall Height (real value)
  - ✓ X6: Orientation (integer)
  - ✓ X7: Glazing Area (real value)
  - ✓ X8: Glazing Area Distribution (integer)
- Targets:
  - ✓ Y1: Heating Load (real value)
  - ✓ Y2: Cooling Load (real value)

III. Preprocess and Split the Data

- Load the dataset (ENB2012_data.xlsx) using pandas.
- Separating features (X1 to X8) and targets (Y1 and Y2).
- Spliting the data into training and testing sets using train_test_split from sklearn.

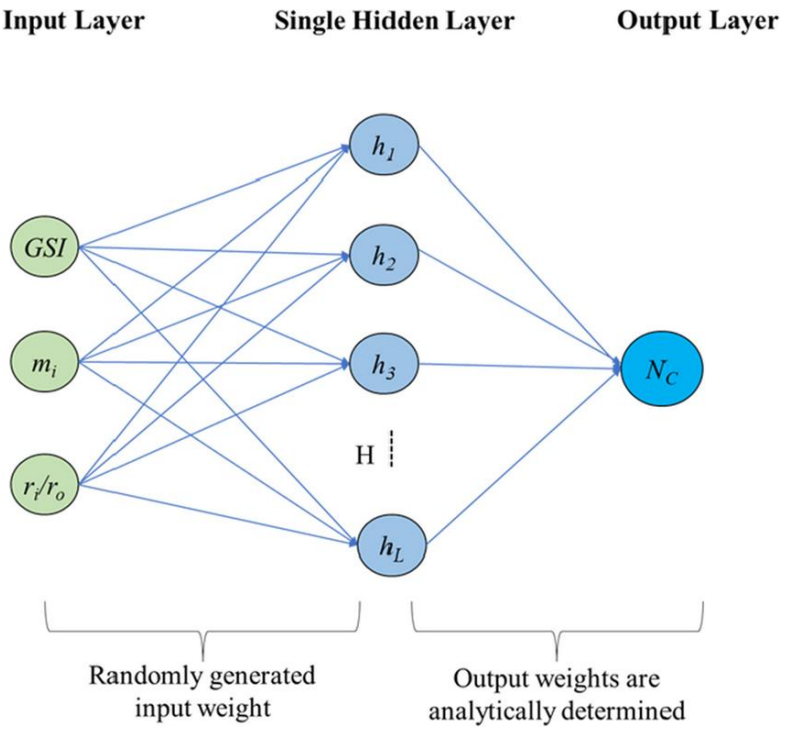
```
from sklearn.model_selection import train_test_split
import pandas as pd

# Load dataset
df = pd.read_excel('ENB2012_data.xlsx')

# Features and targets
X = df.iloc[:, 0:8]  # X1 to X8
y = df.iloc[:, 8:10]  # Y1 and Y2

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

IV. Model Development:

a. Implement ELM Regressor:

What is ELM?

- ELM is a type of single-layer feedforward neural network.
- The input weights and biases are randomly assigned and never updated.
- Only the output weights are learned using linear regression (Moore–Penrose pseudo-inverse).
- It's extremely fast because there's no iterative training like in backpropagation.



Here's an ELM model to predict Heating Load (Y1):

```
import numpy as np
class SimpleELM:
    def __init__(self, input_size, hidden_size, activation=np.tanh, random_state=None):
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.activation = activation

        # Set random seed for reproducibility
        if random_state is not None:
            np.random.seed(random_state)

        # Random input weights and biases
        self.W = np.random.randn(self.input_size, self.hidden_size)
        self.b = np.random.randn(self.hidden_size)

    def fit(self, X, y):
        # Calculate hidden layer output (H)
        H = self.activation(np.dot(X, self.W) + self.b)

        # Calculate output weights (beta) using pseudo-inverse
        self.beta = np.dot(np.linalg.pinv(H), y)

    def predict(self, X):
        H = self.activation(np.dot(X, self.W) + self.b)
        return np.dot(H, self.beta)
```

✅ Using the ELM For Heating Load prediction (Y1):

```
# For Heating Load prediction (Y1)
elm = SimpleELM(input_size=8, hidden_size=100, random_state=42)
elm.fit(X_train.values, y_train.iloc[:, 0].values)  # fit on Y1

y1_pred_elm = elm.predict(X_test.values)

# Evaluate performance
from sklearn.metrics import mean_squared_error
mse_elm = mean_squared_error(y_test.iloc[:, 0], y1_pred_elm)
print("Heating Load MSE (ELM):", mse_elm)
```

Output:

```
Heating Load MSE (ELM): 18.88333995269587
```

What is MSE ?
Mean Squared Error (MSE) is the average of the squared differences between predicted values and actual values. It measures how close a model's predictions are to the true values. **Lower MSE means better model performance.**

Formula

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

✅ Using the ELM For Cooling Load prediction (Y2):

```
# Create a new ELM instance (or reuse the same if you want)
elm_y2 = SimpleELM(input_size=8, hidden_size=100, random_state=42)
elm_y2.fit(X_train.values, y_train.iloc[:, 1].values)  # fit on Y2

# Predict
y2_pred_elm = elm_y2.predict(X_test.values)

# Evaluate
mse_elm_y2 = mean_squared_error(y_test.iloc[:, 1], y2_pred_elm)
print("Cooling Load MSE (ELM):", mse_elm_y2)
```

Output:

```
Cooling Load MSE (ELM): 15.672218893395456
```

b. Implement BP Regressor (Neural Network):

We are going to use MLPRegressor from sklearn.neural_network.

✅ Train BP model for Cooling Load (Y1)

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error

# Train BP model for Heating Load (Y1)
bp_heating = MLPRegressor(hidden_layer_sizes=(50, 50), max_iter=1000, random_state=42)
bp_heating.fit(X_train, y_train.iloc[:, 0])

# Predict and evaluate
y1_pred = bp_heating.predict(X_test)
mse_y1 = mean_squared_error(y_test.iloc[:, 0], y1_pred)
print("Heating Load MSE (BP):", mse_y1)
```

Output:

```
Heating Load MSE (BP): 16.333486414414566
```

✅ Train BP model for Cooling Load (Y2)

```
# Train BP model for Cooling Load (Y2)
bp_cooling = MLPRegressor(hidden_layer_sizes=(50, 50), max_iter=1000, random_state=42)
bp_cooling.fit(X_train, y_train.iloc[:, 1])  # y_train.iloc[:, 1] = Y2 (Cooling Load)

# Predict and evaluate
y2_pred = bp_cooling.predict(X_test)
mse_y2 = mean_squared_error(y_test.iloc[:, 1], y2_pred)
print("Cooling Load MSE (BP):", mse_y2)
```

Output:

```
Cooling Load MSE (BP): 16. 03295696501423
```

V. Model Performance Comparison

🔎 Mean Squared Error (MSE) Recap:
After training both the **Extreme Learning Machine (ELM)** and **Backpropagation (BP)** models, we evaluated their performance using the **Mean Squared Error (MSE)** metric on the test dataset.

Mean Squared Error Results

Target Variable	Model	MSE
Heating Load (Y1)	ELM	❌ 18.88
	BP	✅ 16.33
Cooling Load (Y2)	ELM	✅ 15.67
	B	❌ 16.03

→ Performance Comparison:

- **Heating Load (Y1)**: The **BP** model has a lower MSE (16.33) compared to the **ELM** model (18.88). Therefore, **BP** performs better for predicting Heating Load.
- **Cooling Load (Y2)**: The **ELM** model has a lower MSE (15.67) compared to the **BP** model (16.03). So, **ELM** performs better for predicting Cooling Load.

Conclusion:

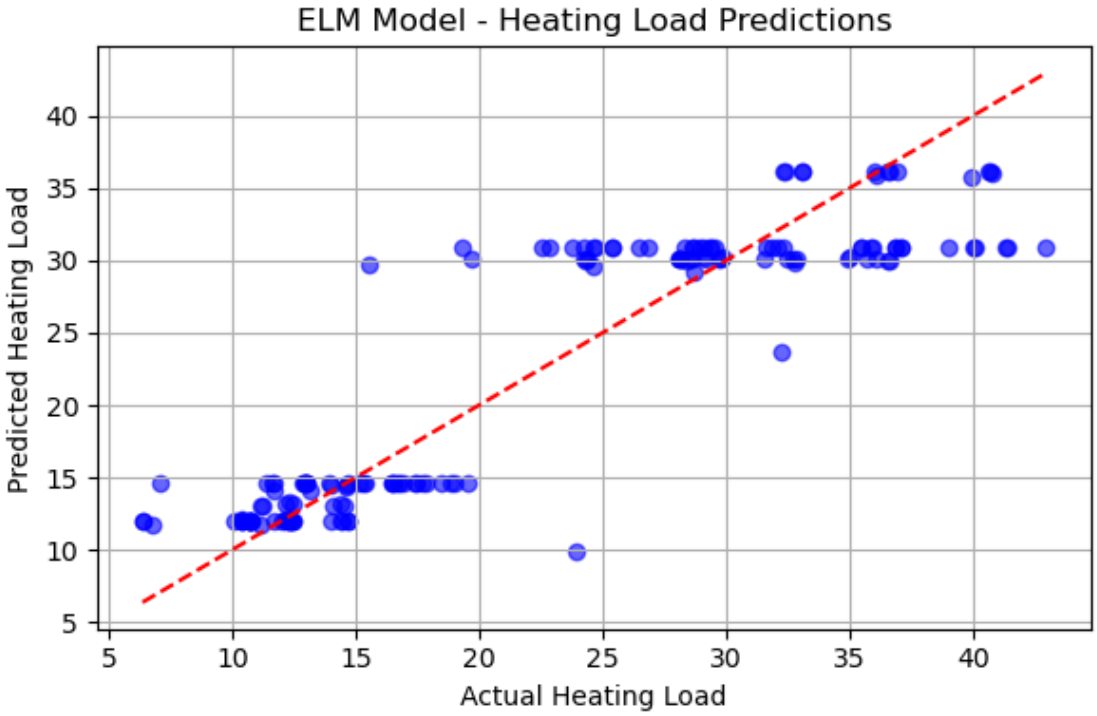
- **For Heating Load (Y1)**, **BP** is the better model.
- **For Cooling Load (Y2)**, **ELM** is the better model.

Note:

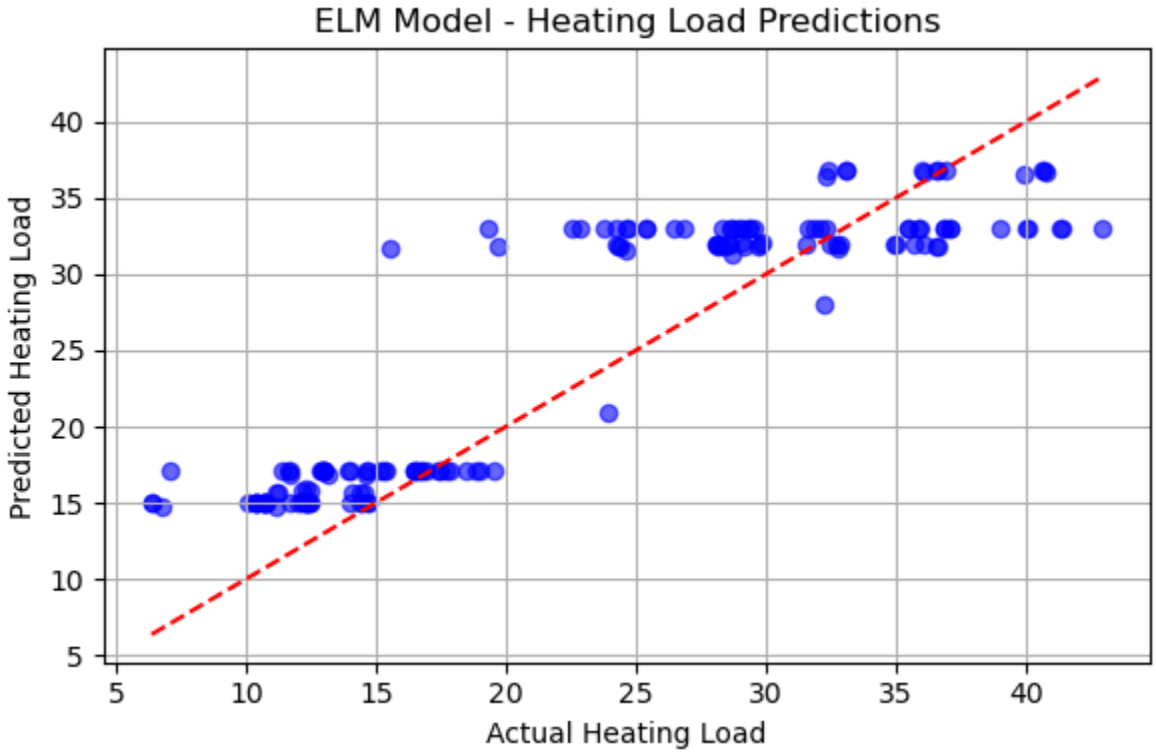
To ensure consistency in the results across multiple runs, we have set the **random_state=42** for the models. This fixed seed ensures that the random processes, such as data splitting and weight initialization, are controlled and reproducible. Without setting the **random_state**, each run could produce different values due to the inherent randomness in processes like shuffling or weight initialization. By fixing the random seed, I guarantee that the model will always receive the same initial conditions, which helps in achieving consistent results and better comparison across different experiments.

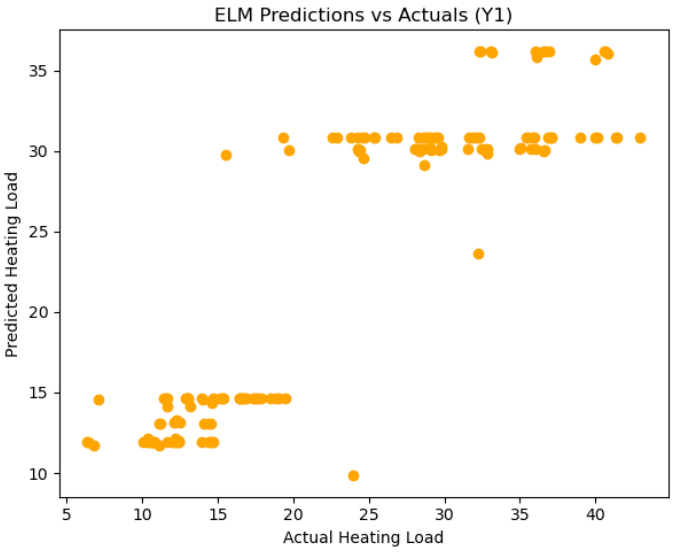
VI. Visualization of Predictions 📈 for ELM

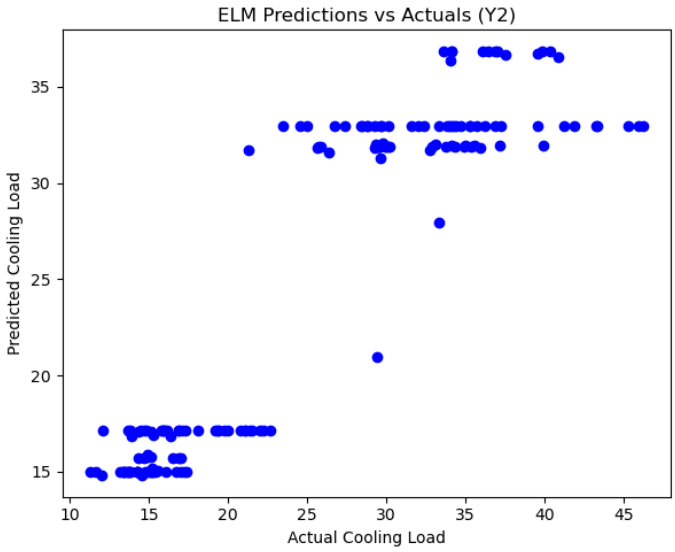
◈ Heating Load (Y1) Predictions – ELM Model:

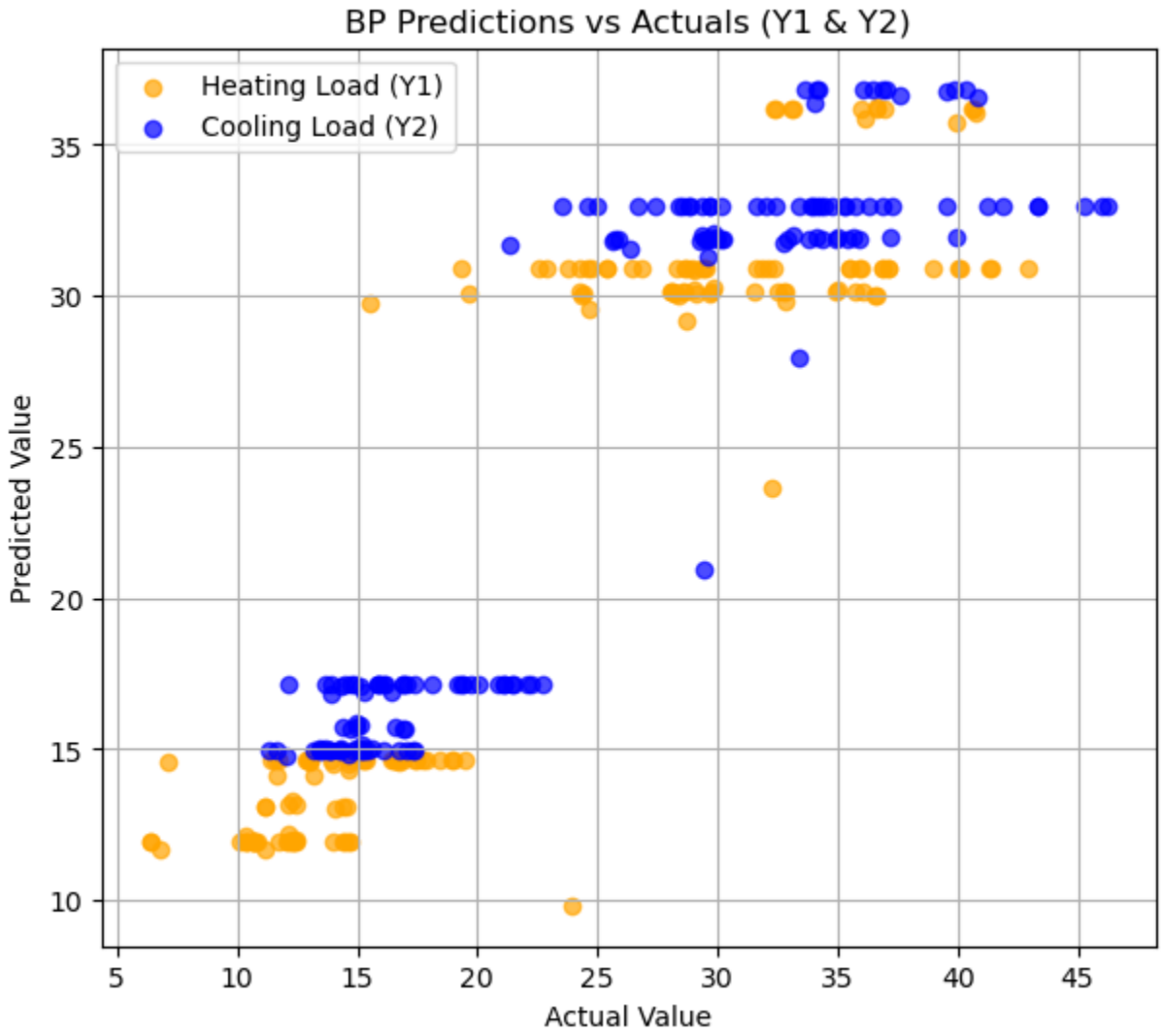


◈ Heating Load (Y2) Predictions – ELM Model:



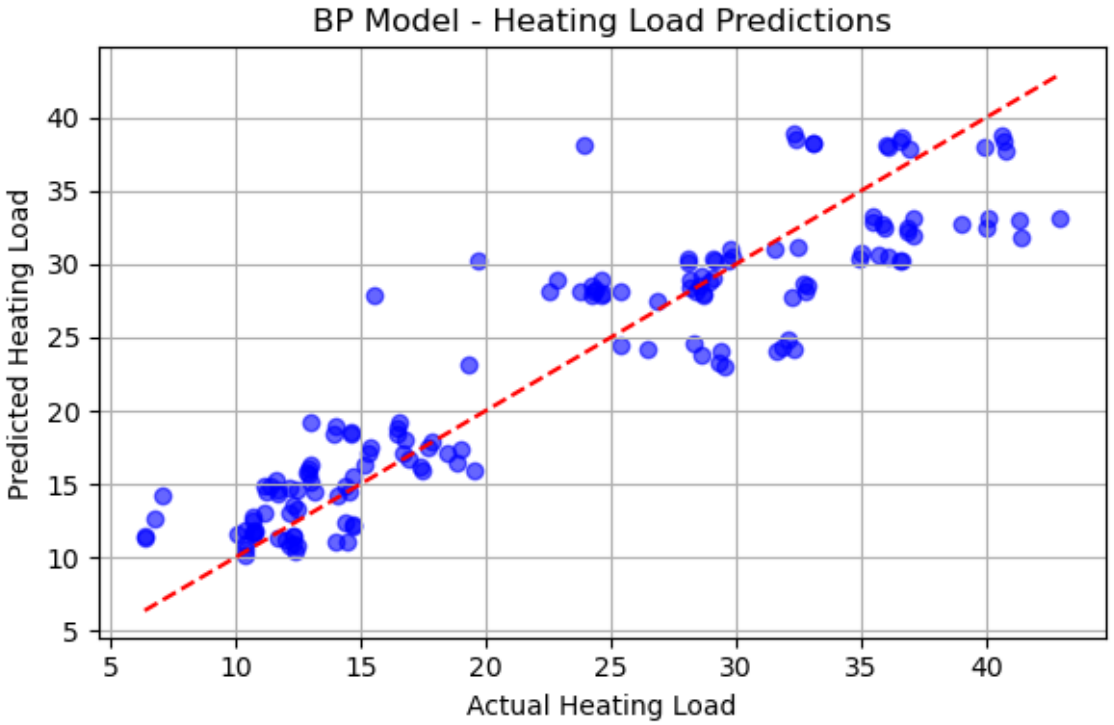

ELM Predictions vs Actuals (Y1)
Predicted Heating Load
Actual Heating Load


ELM Predictions vs Actuals (Y2)
Predicted Cooling Load
Actual Cooling Load

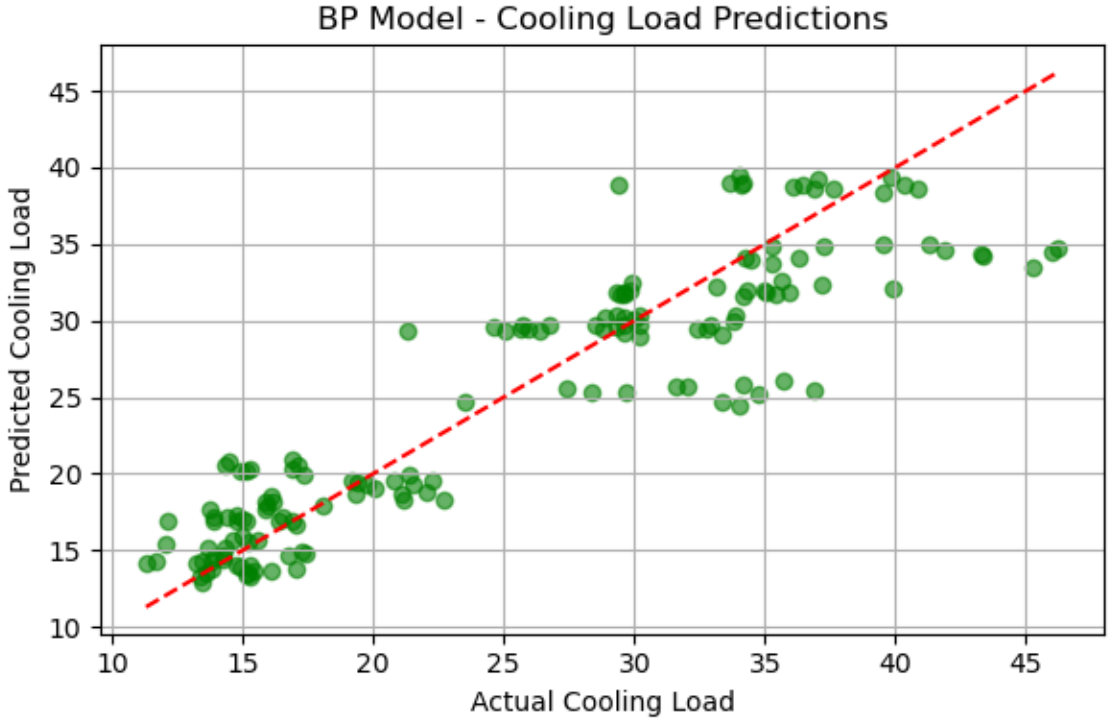

BP Predictions vs Actuals (Y1 & Y2)
Heating Load (Y1)
Cooling Load (Y2)
Predicted Value
Actual Value

VII. Visualization of Predictions 📈 for BP

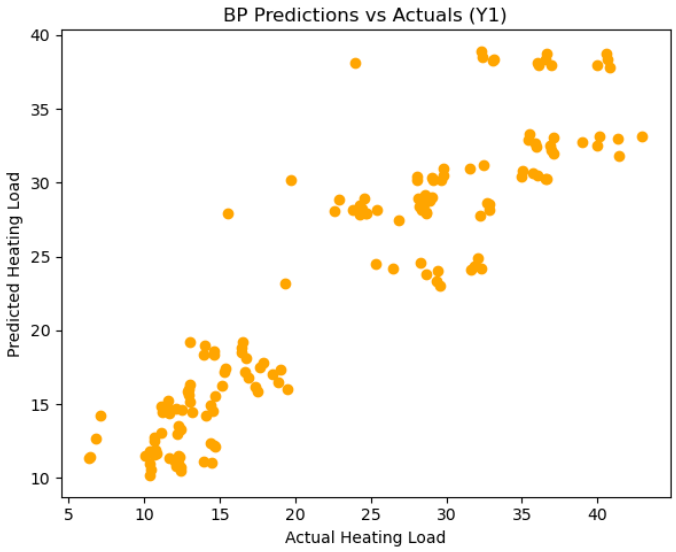
◈ Heating Load (Y1) Predictions – BP Model:

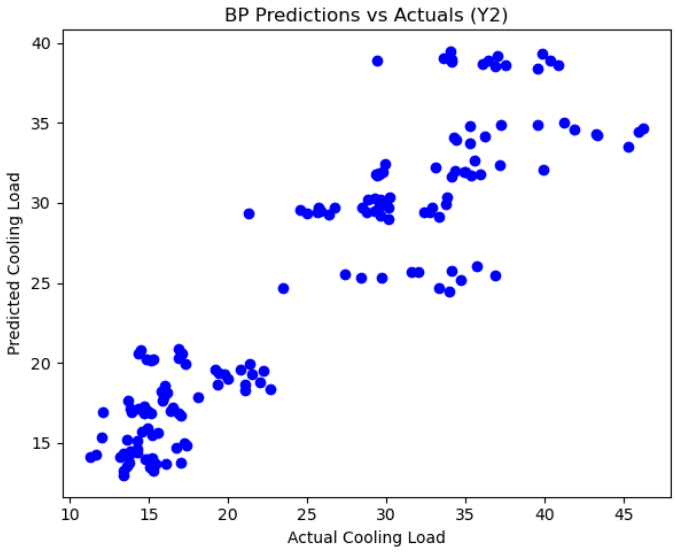


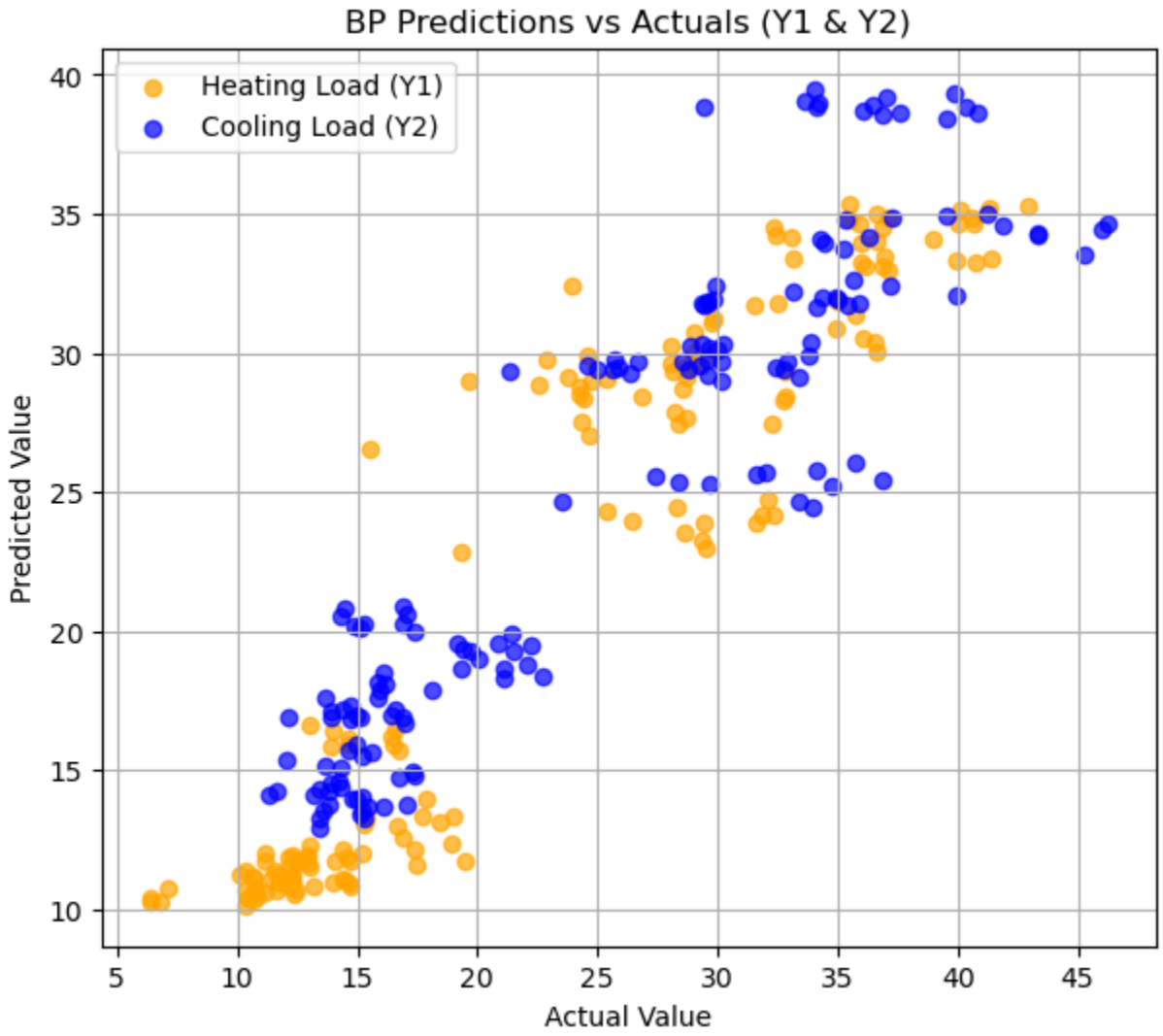
◈ Cooling Load (Y2) Predictions – BP Model:



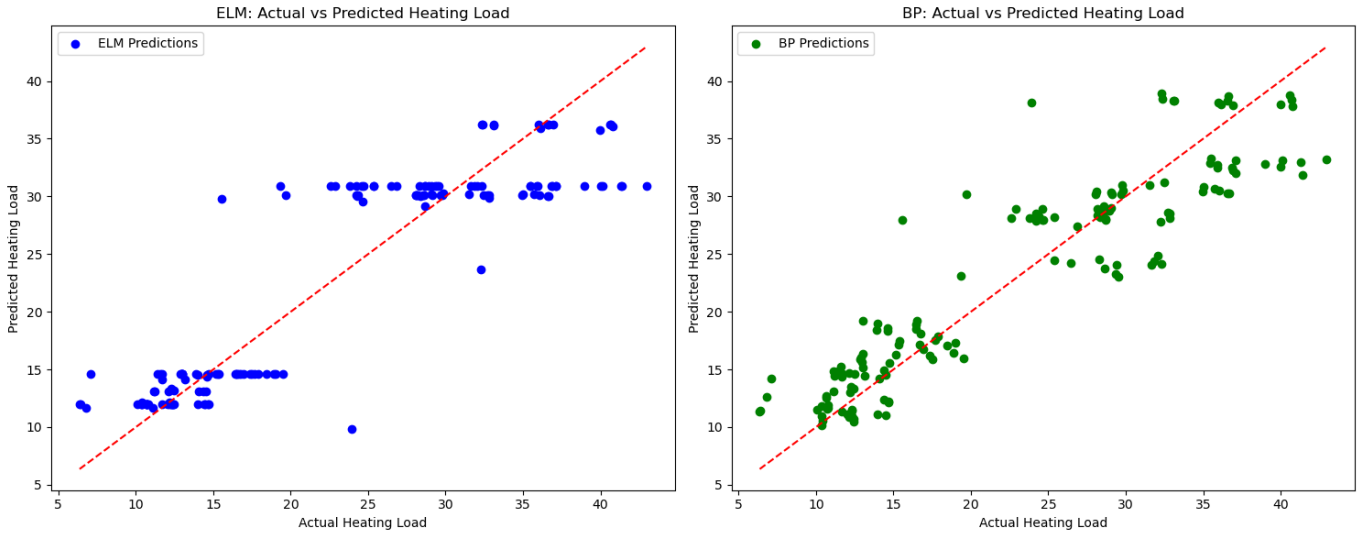
◆ The red dashed line represents the ideal predictions (perfect match between actual and predicted values).


BP Predictions vs Actuals (Y1)
Predicted Heating Load
Actual Heating Load


BP Predictions vs Actuals (Y2)
Predicted Cooling Load
Actual Cooling Load


BP Predictions vs Actuals (Y1 & Y2)
Heating Load (Y1)
Cooling Load (Y2)
Predicted Value
Actual Value

Visualize Heating Load Predictions (Y1) for both BP & ELM:



ELM Model (left plot):

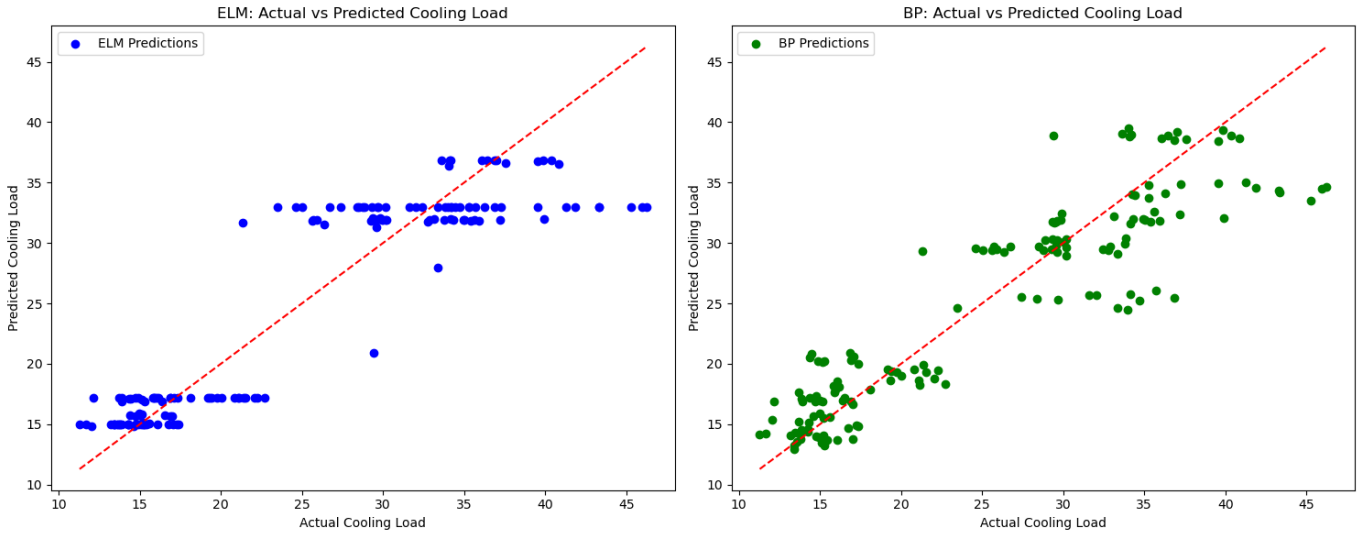
- The data points are clustered in distinct horizontal bands.
- There is limited variation along the predicted axis, indicating that the model outputs a narrow range of values.
- Many points are far from the red dashed line, which represents the perfect prediction, indicating the model's predictions are not very close to the actual values.
- The model appears to underestimate higher heating loads and overestimate lower ones, showing a systematic bias.

BP Model (right plot):

- The data points are more evenly distributed along the diagonal.
- There is a more natural spread in the predictions, suggesting a better range of output values.
- Points are closer to the ideal prediction line (red dashed), indicating better alignment with the actual values.
- While there is still some variance, the BP model's predictions are generally more accurate and track the actual values more closely.

→ The BP model demonstrates better predictive performance, with fewer systematic errors and more accurate alignment with actual values across the range of heating loads. In contrast, the ELM model's outputs seem more quantized, which might suggest limitations in how it captures the underlying relationships between input features and the target variable.

Visualize Cooling Load Predictions (Y2) for both BP & ELM:



ELM Model (left plot):

- The predictions cluster in distinct horizontal bands.
- There is limited variability in the predicted values, suggesting the model is quantizing its outputs to specific values.
- Many predictions deviate significantly from the ideal prediction line (red dashed), indicating poor alignment with the actual values.
- The model does not seem to capture the continuous nature of the cooling load values, leading to less accurate predictions.

BP Model (right plot):

- The data points are more evenly distributed along the diagonal.
- Predictions exhibit a more natural, continuous distribution, better reflecting the actual values.
- Points align more closely with the ideal prediction line, showing better accuracy.
- While some variance exists, there is a clearer correlation between actual and predicted values, demonstrating the model's responsiveness to nuances in the data.

→ Similar to the heating load comparison, the BP model outperforms the ELM model in predicting cooling loads. The BP model exhibits fewer systematic errors and better alignment with actual values across the range, while the ELM model's quantized outputs limit its accuracy by not fully capturing the continuous nature of the cooling load.

VIII. Model Performance Summary ELM vs. BP

The choice between ELM (Extreme Learning Machine) and BP (Backpropagation) largely depends on the specific problem we're trying to solve, the dataset, and other factors, based on that here's a general comparison to help us decide:

ELM (Extreme Learning Machine):

- **Faster Training**: ELM typically has a much faster training time because it randomly initializes the weights in the hidden layer and only needs to compute the output weights via a pseudo-inverse. No iterative training process is required for the hidden layer.
- **Simpler Model**: ELM is simpler to implement, requiring less fine-tuning of hyperparameters compared to BP models.
- **Good for Simple Problems**: It works well for simpler problems with well-behaved data and when you don't require extensive model adjustments or complex feature engineering.
- **Limitations**:
  - It may not generalize well for more complex problems or datasets.
  - It's sensitive to the number of hidden units and can overfit or underfit if not chosen appropriately.
  - The randomness in the weight initialization can lead to varied results across runs, although you can fix this with a fixed random state.

BP (Backpropagation):

- **Flexible and Powerful:** BP (typically used in deep learning with neural networks) is much more flexible and capable of learning complex patterns in large, high-dimensional datasets. It's highly effective in solving difficult, non-linear problems.
- **Better Performance for Complex Problems**: BP is usually more suitable for complex problems, such as image recognition, speech recognition, or tasks with large datasets.
- **Training Can Be Slow**: BP requires iterative optimization (via gradient descent) and can take longer to train, especially with deep networks. It may also be more sensitive to initial weights and hyperparameters, which can require more tuning.
- **Prone to Overfitting**: BP models can easily overfit if not regularized properly, especially when dealing with a small dataset.

Which is Better?

- **For Simplicity and Speed**: ELM may be better for quick experiments, smaller problems, or when speed is crucial.
- **For Complex Problems**: BP (Backpropagation) typically provides better performance for more complex and high-dimensional problems, where deeper models can learn intricate patterns.

Summary:

- **ELM** is fast, easy to implement, and effective for simpler tasks with fewer hyperparameters to tune, but it may struggle with complex problems.
- **BP (Backpropagation)** is more flexible and powerful, especially for complex, high-dimensional problems, but it comes with a heavier computational cost and requires careful tuning.

→ In general, **BP** is more widely used for sophisticated tasks, while **ELM** is a good alternative when you need faster, simpler models.