### **Dataset Overview: Energy Efficiency Dataset**

The **Energy Efficiency Dataset** is derived from the UCI Machine Learning Repository. This dataset is widely used in research to model and analyze energy efficiency in buildings. It contains information about buildings' various physical and operational attributes and their corresponding energy demands.

https://archive.ics.uci.edu/dataset/242/energy+efficiency

### **Features (Inputs):**

The dataset contains **8 numerical features** related to the physical and operational characteristics of a building:

1. **X1 (Relative Compactness)**:
   * The ratio of the volume of the building to its external surface area.

**Indicates how efficiently the building's space is utilized.**

1. **X2 (Surface Area)**:
   * The total external surface area of the building in square meters.
2. **X3 (Wall Area)**:
   * Total area of the building's external walls in square meters.
3. **X4 (Roof Area)**:
   * Total area of the building's roof in square meters.
4. **X5 (Overall Height)**:
   * Height of the building in meters.
5. **X6 (Orientation)**:
   * The cardinal direction the building faces (encoded numerically).
6. **X7 (Glazing Area)**:
   * Percentage of the building’s exterior covered by windows.
7. **X8 (Glazing Area Distribution)**:
   * Distribution of the glazing area (e.g., uniform or concentrated).

**Targets (Outputs):**

There are **two target variables** (both numerical):

1. **Y1 (Heating Load)**:
   * The amount of energy needed to heat the building to a comfortable temperature.
2. **Y2 (Cooling Load)**:
   * The amount of energy needed to cool the building to a comfortable temperature.

For our model, we are focusing on **Y1 (Heating Load)**.

### **The goal of Developing a Machine Learning Model:**

The primary goal of using machine learning with this dataset is to:

#### **1. Predict Energy Efficiency (Heating Load):**

* Develop a model that predicts the **heating load (Y1)** for a given building configuration based on its physical attributes.
* This helps architects, engineers, and energy specialists evaluate building designs in terms of energy efficiency without performing costly simulations or physical tests.

#### **2. Understand Feature Impact:**

* Identify which features (e.g., compactness, surface area, glazing area) have the most significant influence on energy demands.
* This aids in optimizing building designs for maximum energy efficiency.

#### **3. Improve Sustainability:**

* By predicting energy demands accurately, we can explore ways to design buildings that reduce energy consumption, minimize costs, and lower carbon footprints.

#### **4. Optimize Energy Resource Allocation:**

* Efficiently allocate energy resources by understanding the expected energy demands of buildings under different conditions.

#### **5. Facilitate Data-Driven Decision-Making:**

* Assist policymakers and building planners with data-driven insights to enforce energy-efficient standards in urban planning and construction.

### **Machine Learning Approach:**

1. **Supervised Learning Task**:
   * The dataset is labeled with output variables (Y1 and Y2), making this a supervised learning regression problem.
2. **Target Variable**:
   * We focus on predicting **Y1 (Heating Load)**, which is continuous.
3. **Model Objectives**:
   * Build a regression model (e.g., Random Forest, Linear Regression, etc.) to minimize the prediction error (e.g., RMSE).
   * Use insights from the model to prioritize features in the building design process.
4. **Evaluation Metrics**:
   * **Root Mean Squared Error (RMSE)**: Measures the model's prediction accuracy.
   * **Mean Absolute Error (MAE)**: Evaluates the average prediction error.
   * **R² (Coefficient of Determination)**: Determines how well the model explains the variance in the data.

By achieving these goals, the machine learning model contributes to more sustainable and efficient building designs, reducing energy waste and promoting environmental conservation.

IN SECOND CODE:

To increase the accuracy of your model (already performing well with an R² of ~0.997), you can refine the preprocessing steps, optimize the model, and experiment with alternative techniques. Below are specific strategies:

### **1. Feature Engineering**

* **Polynomial Features**: Include interaction terms or polynomial features to capture non-linear relationships.

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2, interaction\_only=True, include\_bias=False)

X\_poly = poly.fit\_transform(X)

X = pd.DataFrame(X\_poly)

* **Domain-Specific Transformations**: If domain knowledge suggests certain transformations, such as logarithmic or square-root scaling, apply them.
* **Feature Selection**: Use feature importance (e.g., from Random Forest or mutual information) to select the most impactful features.

### **2. Hyperparameter Tuning**

Optimize the parameters of your Random Forest model using techniques like Grid Search or Random Search.

#### **Grid Search:**

from sklearn.model\_selection import GridSearchCV

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

}

grid\_search = GridSearchCV(estimator=RandomForestRegressor(random\_state=42),

param\_grid=param\_grid,

scoring='neg\_mean\_squared\_error',

cv=5,

verbose=2)

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

**Random Search**:

from sklearn.model\_selection import RandomizedSearchCV

random\_grid = {

'n\_estimators': [int(x) for x in np.linspace(start=100, stop=1000, num=10)],

'max\_features': ['auto', 'sqrt'],

'max\_depth': [None] + [int(x) for x in np.linspace(10, 100, 10)],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

}

random\_search = RandomizedSearchCV(estimator=RandomForestRegressor(random\_state=42),

param\_distributions=random\_grid,

n\_iter=100, cv=3, verbose=2, random\_state=42,

scoring='neg\_mean\_squared\_error')

random\_search.fit(X\_train, y\_train)

best\_model = random\_search.best\_estimator\_

### **3. Try Alternative Algorithms**

Experiment with other regression models that may provide better accuracy for your dataset:

* **Gradient Boosting Models**: Like XGBoost, LightGBM, or CatBoost.

from xgboost import XGBRegressor

model = XGBRegressor(n\_estimators=100, learning\_rate=0.1, max\_depth=5, random\_state=42)

model.fit(X\_train, y\_train)

* **Support Vector Regression (SVR)**: Effective for non-linear relationships.

from sklearn.svm import SVR

model = SVR(kernel='rbf', C=100, epsilon=0.1)

model.fit(X\_train, y\_train)

* **Deep Learning**: Use neural networks for complex patterns

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(64, input\_dim=X\_train.shape[1], activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')

model.fit(X\_train, y\_train, epochs=100, batch\_size=32)

### **4. Address Overfitting or Underfitting**

* **Regularization**: If your model overfits, apply regularization to prevent it from memorizing noise.
  + For Random Forest, adjust max\_depth, min\_samples\_split, or min\_samples\_leaf.
* **Cross-Validation**: Use k-fold cross-validation to ensure the model generalizes well.

### **5. Use Ensembles**

Combine predictions from multiple models to reduce variance and improve accuracy.

from sklearn.ensemble import VotingRegressor

model1 = RandomForestRegressor(n\_estimators=200, random\_state=42)

model2 = XGBRegressor(n\_estimators=100, learning\_rate=0.1, max\_depth=5, random\_state=42)

ensemble = VotingRegressor(estimators=[('rf', model1), ('xgb', model2)])

ensemble.fit(X\_train, y\_train)

### **6. Improve Data Quality**

* **Augment Data**: Create synthetic samples using techniques like SMOTE for imbalance or perturbing existing samples.
* **Remove Noise**: Outliers or noisy data points can distort the model. Ensure robust preprocessing.
* **Increase Data Volume**: If feasible, gather more data to allow the model to learn better patterns.

### **7. Explore Feature Selection Techniques**

Use algorithms like Recursive Feature Elimination (RFE) to remove irrelevant features.

from sklearn.feature\_selection import RFE

selector = RFE(RandomForestRegressor(random\_state=42), n\_features\_to\_select=5)

X\_selected = selector.fit\_transform(X, y)

### **8. Advanced Techniques**

* **Bayesian Optimization**: Automate hyperparameter tuning for better results.
* **Transfer Learning**: Use pre-trained models if similar datasets are available.
* **AutoML**: Tools like H2O AutoML or Google AutoML can help find optimal models and hyperparameters.

### **Evaluation:** Once modifications are made, evaluate the model again using metrics like RMSE, MAE, and R². Compare results with the baseline model to assess improvements.

These steps will help you fine-tune your model, explore alternative approaches, and ensure better performance and accuracy.