NAME: SOUMIT SAHA

COLLEGE: INDIAN INSTITUTE OF TECHNOLOGY, PATNA

APPLIED FOR: DATA SCIENCE INTERNSHIP

MECHADEMY: SMART FACTORY ENERGY PREDICTION CHALLENGE

DOCUMENTATION

This document outlines the sequential steps and methods implemented in the notebook to build a predictive model for equipment energy consumption.
The project aims to develop a machine learning regression model to predict the energy consumption of industrial equipment using sensor data collected from a factory's operational environment. The objective is to uncover patterns and correlations between environmental variables and equipment energy usage, enabling facility managers to make data-driven decisions that enhance energy efficiency and reduce costs. The approach includes comprehensive data preprocessing, model training using advanced algorithms, and performance evaluation using metrics such as RMSE, MAE, and R^2. Based on the findings, actionable insights and recommendations will be provided to optimize equipment operations and lower overall energy consumption.

CONTENTS:

1) IMPORTING REQUIRED LIBRARIES

- Pandas, Numpy for data handling
- matplotlib, seaborn for visualization
- sklearn modules: train_test_split, ParameterGrid, StandardScaler, error metrics, RandomForestRegressor

2)LOADING THE DATASET

- Loaded data.csv and displayed head i.e the first 5 rows of the given dataset.

3)INITIAL DATA INSPECTION

- I Checked record count and missing values

4) HANDLING MISSING VALUES

- Inspected missing value counts per column and then implemented an imputation strategy:
  - For numeric columns, replaced missing entries with the **mean** of the column.
  - For categorical columns, filled missing entries with the **mode** of the column.

5) TARGET TRANSFORMATION

- Applied **log1p** transform to target.
  The Reason for applying the ***logarithmic transformation***:

1. ***Reduces Skewness*** A right-skewed distribution has a long tail on the right (many small values, few very large ones). Taking the logarithm compresses large values and stretches small ones, making the distribution more symmetric and closer to normal.
2. ***Improves Model Performance*****:** Many regression models (especially linear models and tree-based models) perform better when the target variable is more normally distributed. It helps the model better understand the relationship between inputs and output, especially when changes in the target variable are multiplicative rather than additive.
3. ***Reduces Impact of Outliers*****:** In raw form, large values can disproportionately influence the model. Log transformation dampens outlier impact without completely discarding the information.
4. ***Makes Errors More Interpretable*****:** In log space, the model's prediction error corresponds to a relative (percentage-like) difference rather than an absolute one.

6) FEATURE ENGINEERING & PREPROCESSING AND EXPLORATORY DATA ANALYSIS (EDA)

- Assessed target distribution, converted timestamp, calculated correlations, plotted scatter plots.

- Selected predictors, handled categorical features, split and scaled data

1. **Selecting Predictor Variables**

    - Dropped the target column equipment energy consumption and non-informative columns: timestamp, random_variable1, and random_variable2.
    - Resulting feature matrix X contains only sensor and environmental variables expected to influence energy consumption.

2. **Handling Categorical Features**

    - Identified any columns with dtype object.
    - Replaced 'unknown' strings with NaN for consistency.
    - Imputed missing values using the column mode so categorical distributions remain representative.
    - Converted all categorical columns to numeric codes to prepare for modelling.

3. **Train–Test Split**

    - Split the data into training and testing sets to evaluate model generalization.
    - Used an **80/20 split** with a fixed **random_state=42** for reproducibility.

4. **Feature Scaling**

    - Applied **StandardScaler** to **centre** features at **zero mean** and **unit variance**, which benefits tree-based models less but ensures compatibility if switching to algorithms sensitive to feature scales.
    - Fit the scaler on the training data and applied the same transformation to the test set.

-INTERPRETATIONS OF THE DISTRIBUTION OF THE RESIDUAL

- **Peak near 0:**
  Most of the predictions are close to the actual values.
- **Slight right skew:**
  There are more large positive errors than negative ones: this suggests **that the model slightly underpredicts** energy usage in some cases.
- **Long tail on the right:**
  A few predictions are significantly lower than the actual values: these are **outliers** where the model missed the true value by a large margin.

7) MODEL DEVELOPMENT & HYPERPARAMETER TUNING

- To build the regression model, I selected **RandomForestRegressor** for its ability to capture nonlinear relationships and handle feature interactions without extensive parameter tuning.

Without **Cross-Validation:**

1. **Defining the Hyperparameter Grid**
   - We created a dictionary of hyperparameters to explore:
     - **n_estimators**: Number of trees in the forest (tested 100 and 200).
     - **max_depth**: Maximum depth of each tree (None for unlimited growth, and fixed depths 10 and 20 to prevent overfitting).
   - Using a concise grid allows systematic exploration while keeping computational cost manageable.
2. **Manual Grid Search with ParameterGrid**
   - **ParameterGrid** generates all combinations of hyperparameters.
   - For each combination:
     - Instantiate a RandomForestRegressor with the given parameters and random_state=42 for reproducibility.
     - Fit the model on the **training set** (X_train_scaled, y_train).
     - Predict on the **validation set** (X_test_scaled) and compute the **Root Mean Squared Error (RMSE)**.
   - Track the model achieving the lowest RMSE as the **best model**.

With **Cross-Validation**:
Defining the Hyperparameters grid:

1. n_estimators :

- **Definition**: The number of trees in the random forest.
- **Impact**: I have used this as more **trees usually improve performance** but **increase training time**.
- **Values**:
    - 100: Standard starting point.
    - 200: Doubles the number of trees for potentially better ensemble accuracy.

2. max_depth :

- **Definition**: Maximum depth of each tree.
- **Impact**:
    - Smaller depth → more **regularization** (less overfitting).
    - Larger depth → more complex models (risk of overfitting).
- **Values**:
    - 10: This will limit tree depth to avoid **overfitting**.
    - None: I have used this because it allows trees to expand until all leaves are pure or contain fewer samples than min_samples_split.

3. min_samples_split :

- **Definition**: Minimum number of samples required to split an internal node.
- **Impact**: Higher values prevent the model from learning overly specific patterns.
- **Value**: 2 (default) — allows splits as long as a node has 2 or more samples.

4. min_samples_leaf :

- **Definition**: Minimum number of samples required to be at a leaf node.
- **Impact**:
    - Higher values → smoother predictions (regularization).
    - Lower values → potentially more complex trees.
- **Value**: 1: I had chosen this after a lot of experiments and found out that this allows leaves to have a single data point (more flexible).

5. max_features

- **Definition**: The number of features to consider when looking for the best split.
- **Impact**: Controls randomness and diversity in individual trees.
- **Value**:
    - 'sqrt': This uses the square root of the total number of features. I have used this to reduce overfitting and improve generalization.

8) SELECTION AND JUSTIFICATION OF ALGORITHM

I selected the **Random Forest Regressor** because it is a powerful **ensemble method** that **performs well on tabular sensor data**. It **captures non-linear patterns effectively** and is **less prone to overfitting due to its use of multiple decision trees and averaging**. This makes it well-suited for predicting energy consumption in the presence of diverse and possibly noisy environmental features. Additionally, Random Forests provide a natural way to estimate feature importance, aiding in model interpretation.

Further I have used **RandomizedSearchCV** to efficiently tune the hyperparameters and because **RandomizedSearchCV** also helps in **efficient sampling.**

I have also implemented K-Fold Cross Validation to see how my model performs with and without it. I have used K = 3 here, so I have split my training into 3 parts. Each fold is used once as a validation set while the remaining 2 folds are used for training.

9) MODEL EVALUATION

- -Predicted on test set, evaluated RMSE, Mean Absolute Error and R^2 Score using Cross-Validation:

Metrics	Score
Best CV RMSE	0.106
Test RMSE	0.105
Test MAE	0.051
Test R^2	0.938

10)MODEL LIMITATIONS

1. MEAN-LEVEL BIAS IS SMALL BUT REAL

- We can see that the two mean-confidence intervals overlap heavily, the model's average prediction (≈ 64.5 Wh) is in the same ballpark as the true average (≈ 65.8 Wh).
- **However**, the predicted-mean confidence-interval is shifted **slightly downward**. This tells that the model has a **small negative bias** on average , it tends to underpredict by about ~1 Wh.

2. PREDICTED RANGE IS TOO "NARROW"

- The actual data span from **20 to 140 Wh**, but the model's **95 % predicted interval** only covers **29 to 122 Wh**.

- In practice this means:
    - **Very low** consumption days (< 29 Wh) almost never get predicted.
    - **Very high** peaks (> 122 Wh) are systematically **underestimated**.
- This "shrinkage" toward the mean is a sign of regression toward the mean (or the model choosing a conservative central estimate.)

11) RESULTS VISUALIZATION

- Tabular comparison and scatter plot of actual vs. predicted

Observations:

From the following graphs, I saw that the confidence interval in the actual graph is wider as compared to the predicted distribution plot.

Observations:

1. **High spikes in early 2016 (Jan–May)** consistently reach over **1000 units**.
2. **Long data gaps (May to September)** — possibly due to:
    - Missing data
    - Machine shutdown
    - Sensor failure
3. **A few short bursts in Oct/Nov**, but with **much lower peak values**.

I have separately plotted the histograms for the actual values of the equipment energy consumption as well as the predicted equipment energy consumption.

12) SUGGESTIONS TO DECREASE PEAK ENERGY CONSUMPTION:

1) Load Balancing

- **Shift tasks** from high-load periods (e.g., early morning or noon) to **low-demand periods** (late night).
- Stagger operation of heavy machinery rather than running them all at once.

2) Detect High-Peak Causes

- Check logs to correlate **what processes ran on days with peaks >1000**.
- Investigate which equipment or combination is responsible.

3) Preventive Maintenance

- High energy spikes can signal **inefficiency or wear** in machines (e.g., motors consuming more).
- Regular servicing can lower baseline consumption and smoothen peaks.

4) Install Smart Monitoring

- Use real-time monitoring to trigger alerts if energy usage exceeds a defined threshold.
- Implement automated shutdown or throttling if spikes persist.

5) Energy-Efficient Equipment

- Identify older equipment contributing to spikes and assess **energy-efficient replacements**.

6) Optimize Process Scheduling

- If possible, reschedule energy-intensive jobs to **spread load** more evenly through the day/week.

7)Anomaly Detection Automation

- Use ML models to detect anomalies/spikes **before** they escalate.
- You can implement a moving average or rolling standard deviation to flag sudden jumps.

13) CONCLUSION

In this project, I developed a machine learning model to predict equipment energy consumption using environmental and sensor data from a manufacturing facility. After preprocessing and hyperparameter tuning, the Random Forest model demonstrated strong predictive performance. This solution can assist facility managers in identifying energy-intensive conditions and making informed decisions to improve energy efficiency and reduce costs.