Smart Factory Energy Prediction Challenge

-Mahalakshmi M

-mahalakshmimageswaran@gmail.com

Problem overview: The task is to analyse factory sensor data to build a machine learning model that accurately predicts equipment energy consumption, ultimately providing insights for cost reduction.

NOTE: Please look at main.py(python script) as the main scripts where else notebook (DS-Intern-Smart Factory Energy Prediction.ipynb) has only basic structure and limited functions compared to main.py .

- **The Data:** Sensor readings from a factory, including environmental factors and zone-specific data, stored in data/data.csv. A detailed explanation of each data point is in docs/data_description.md.
- **The Challenge:** Identify which sensor readings are important for predicting energy use, including deciding whether random_variable1 and random_variable2 are useful.
- **The Repository:** Organized with a data folder for the dataset and a docs folder for the data description.
- **My Tasks:**
    1. Explore the data to find relationships.
    2. Build a machine learning model to predict energy use.
    3. Check how well the model predicts.
    4. Suggest ways to reduce energy consumption.

Models used and justifications:

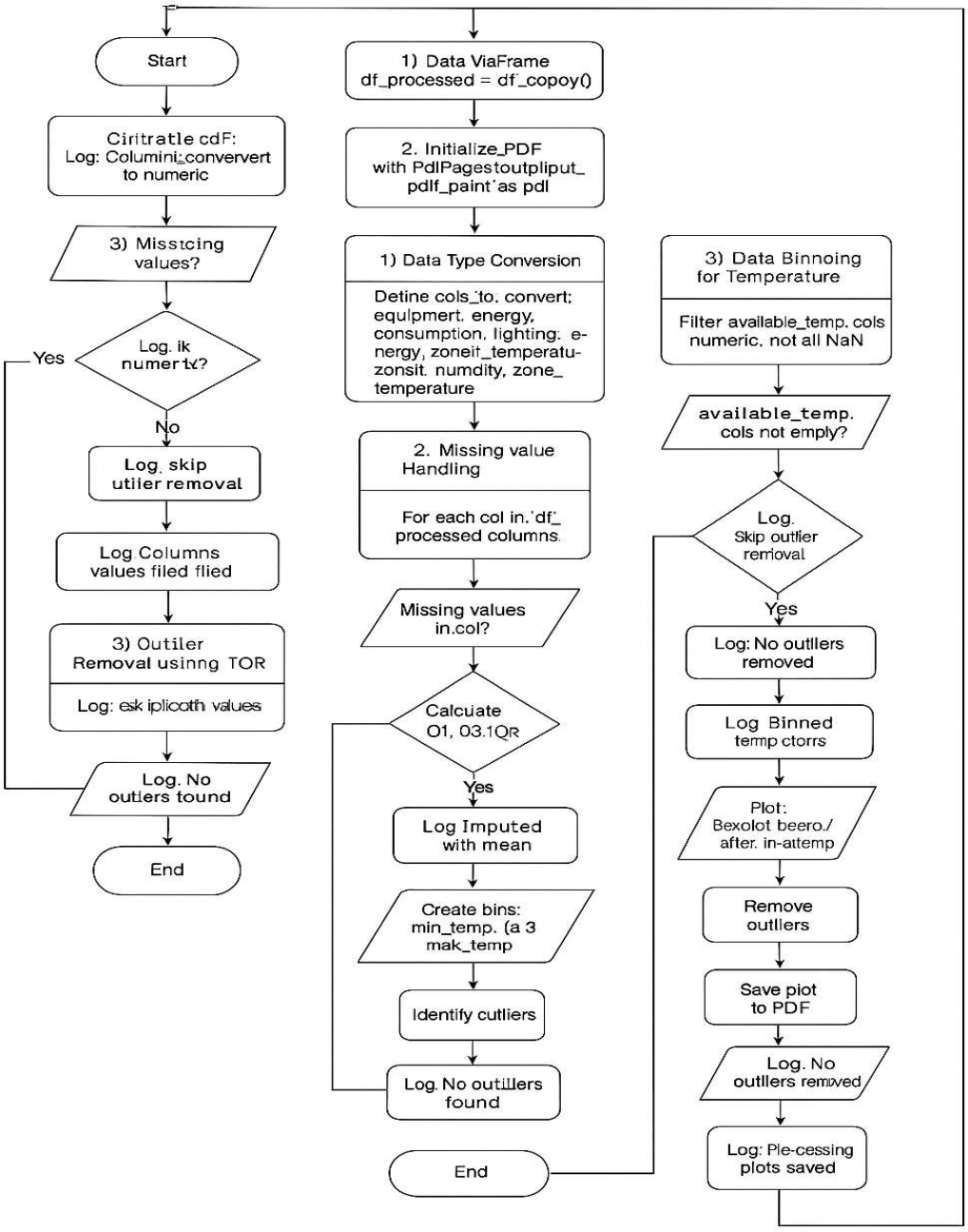
- **Models:**

  1. Random Forest: An ensemble method that averages predictions from multiple decision trees. It handles non-linearity, reduces overfitting, and provides feature importance.

2. **Gradient Boosting:** An ensemble method that sequentially combines weak learners (decision trees) to improve prediction accuracy. It's robust and can capture complex relationships.

- **Hyperparameter Tuning:** This optimizes model performance by finding the best settings (e.g., tree depth) via Grid Search.
- **Cross-Validation:** This assesses model generalization by evaluating it on multiple data subsets, providing a more reliable performance estimate.

FLOWCHART



1. UNDERSTANDING THE DATASET

- Data frame Dimensions: The dataset contains 16,857 rows and 29 columns.
- Mixed Data Types: Several columns that likely contain numerical data (equipment_energy_consumption, lighting_energy, zone temperatures/humidities) are currently stored as the object data type and need conversion. The timestamp column is also an object and should be converted to datetime.
- Missing Values Present: The Non-Null Count in the DataFrame information is less than the total number of entries for many columns, indicating the presence of missing values that will need to be addressed.
- Potential Data Issues: The summary statistics reveal unusual minimum and maximum values (e.g., negative humidity and temperatures), suggesting the need for data cleaning or further investigation into the validity of these readings.
- Timestamp for Time Series Analysis: The presence of a timestamp column indicates that this dataset is likely time-series data, which can be leveraged for temporal analysis and modelling.

2. DATA PREPROCESSING (preprocess_data):

Takes a Data Frame and an output path for PDF plots as input.
Performs several data cleaning and transformation steps.

Key Processing Steps:

- **Data Type Conversion:**Converts the 'timestamp' column to datetime format.
- Converts specified columns ('equipment_energy_consumption', 'lighting_energy', 'zone1_temperature', 'zone1_humidity', and 'zone2_temperature') to numeric, coercing errors to NaN.
- **Missing Value Handling:**Imputes missing values in numeric columns with the mean.
- Imputes missing values in non-numeric columns with the mode.
- **Outlier Removal:** Removes outliers from specified numeric columns using the IQR method.
- Plots the data before and after outlier removal and saves the plots to a PDF.
- **Temperature Binning**: Bins temperature columns into categories ('low', 'mid', 'high') based on predefined thresholds (18 and 25).

Insights:

Here are insights and understandings from the graphs:

- **equipment_energy_consumption**: Shows a significant reduction in the range of energy consumption values after outlier removal. (Page 1)
- **lighting energy**: Demonstrates a compressed range after processing, with the removal of outliers. (Page 2)

- **zone1_temperature**: Displays a slight narrowing of the temperature range after outlier treatment. (Page 3)
- **zone1_humidity**: Illustrates the removal of humidity outliers, resulting in a more concentrated distribution. (Page 4)
- **zone2_temperature**: Shows a small reduction in the temperature range after outlier removal. (Page 5)
- **outdoor temperature**: Depicts the trimming of outdoor temperature outliers, leading to a more central temperature distribution. (Page 6)
- **atmospheric pressure**: Shows the removal of distinct outlier points, correcting for extreme pressure readings. (Page 7)
- **outdoor humidity**: Illustrates the removal of very low outdoor humidity outliers. (Page 8)
- **windspeed**: Displays the removal of both low and high wind speed outliers. (Page 9)
- **visibility index**: Shows a narrowed range after outlier removal, indicating more consistent measurements. (Page 10)
- **dewpoint**: Depicts the removal of dew point outliers. (Page 11)
- **random_variable1**: Shows a reduced spread of values after outlier removal. (Page 12)
- **random_variable2**: Illustrates a decrease in the range of values after processing. (Page 13)

3. EDA

- **Initialization**: The function sets up the EDA process by creating a copy of the input data, initializing a plot counter, and opening a PDF file to save the generated plots.
- **Time Series Analysis**: If available, it generates a time series plot of energy consumption and performs seasonal decomposition to analyze trends and patterns.
- **Correlation Analysis:** It calculates and visualizes the correlation between numerical features using a heatmap, and also identifies potential multicollinearity.
- **Scatter Plots:** The function creates scatter plots of energy consumption against key predictor variables to visualize their relationships.
- **Output:** Finally, it logs the number of generated plots, the PDF file path, and returns the original, unmodified DataFrame.

Insights:

Here are insights and understandings from the graphs:

- **Time Series Plot (Page 1):**

1. Energy consumption shows high variability in the first half of 2016, with frequent spikes.

2. A significant drop in energy consumption occurs around May 2016, and it remains relatively low and stable for the rest of the year.

- **Seasonal Decomposition (Page 2):**

3. The time series has a clear upward trend.
4. The seasonal component shows a repeating pattern, indicating daily or weekly fluctuations in energy consumption.
5. The residual component shows the variations in energy consumption that are not explained by the trend or seasonality.

- **Correlation Heatmap (Page 3):**

6. The heatmap shows the correlations between different variables.
7. 'equipment_energy_consumption' has a positive correlation with 'zone1_temperature'.

- **Scatter Plots (Pages 4-6):**

8. The scatter plots illustrate the relationship between energy consumption and other variables, such as lighting energy and zone temperatures, showing how energy consumption varies as these factors change.

4. FEATURE ENGINEERING

- **Initialization**: The function copies the input DataFrame and prepares to track new features.
- **Time-Based Features**: Extracts hour, day, and month from the 'timestamp' column, if it exists.
- **Interaction Features**: Creates interaction terms between temperature columns and 'hour_of_day'.
- **Lagged and Rolling Features**: Calculates 24-hour lagged and rolling mean for 'equipment_energy_consumption', filling missing values with the mean.
- **Final Adjustments**: Drops the original 'timestamp' column, converts binned categories to numerical codes, fills remaining missing numerical values with the mean, and returns the modified DataFrame.

Insights:

Here are the important insights from the feature engineering output:

- **Time-Based Features Added**: The model now includes hour of day, day of the week, and month, which is crucial for capturing how energy consumption varies with time.
- **Temperature Interactions Considered**: The interaction between temperature and hour of day was included. This shows that the impact of temperature on energy consumption is not constant but changes throughout the day.
- **Lagged and Rolling Energy Data Used**: The inclusion of 24-hour lagged and rolling average energy consumption indicates that the model will account for daily patterns and recent energy trends.

5. FEATURE SELECTION

- Handling missing data.
- Splitting data for evaluation.
- Calculating feature importance using Permutation Importance with a Random Forest.
- Returning a DataFrame with only the selected features.

Insights:

Here are the important insights from the feature engineering graph:

- **Key Influencers**: The graph clearly shows that 'energy_rolling_24' is by far the most important feature.
- **Temperature Interactions Matter**: Several temperature-related interaction features (e.g., 'zone3_temperature_x_hour', 'zone4_temperature_x_hour', 'zone1_temperature_x_hour', and 'zone8_temperature_x_hour') are among the top contributors. This highlights that the effect of temperature on energy consumption varies by hour.
- **Time of Day is Important**: The 'hour_of_day' is also a significant predictor.
- **Other Factors**: 'Atmospheric pressure' also plays a role in predicting 'equipment_energy_consumption'.

6. TRAIN AND EVALUATE MODEL

explanation of how the code works:

- **Model Training and Evaluation**: The code defines a function train_and_evaluate_model that trains and evaluates a regression model. It can use either a Random Forest or a Gradient Boosting model.
- **Data Splitting**: The input data (X and y) is split into training and testing sets.
- **Model Selection and Configuration**:
    - For Random Forest, it sets up a parameter grid for n_estimators, max_depth, and min_samples_split.

- For Gradient Boosting, it sets up a grid for n_estimators, learning_rate, and max_depth.

- **Cross-Validation and Grid Search**: The code performs cross-validation to assess the model's performance and uses Grid Search to find the best hyperparameters.
- **Evaluation and Output**: The code evaluates the best model on the test set using R2, MSE, MAE, and RMSE. It then logs the cross-validation results (before and after tuning) and the test set performance metrics.

Insights:

Here are the important insights from the feature engineering output:

- **Tuning Works**: Hyperparameter tuning improves model performance.
- **Random Forest is better**: Random Forest performs slightly better than Gradient Boosting.
- **Performance:** Both models explain roughly 50% of the data variance.
- **Error**: Both models have similar prediction errors.
- **Parameters**: The best models have specific configurations for tree count, depth, and split criteria.

Metric	Random Forest (Before Tuning)	Random Forest (After Tuning)	Random Forest (Test Set)	Gradient Boosting (Before Tuning)	Gradient Boosting (After Tuning)	Gradient Boosting (Test Set)
R2 (mean)	0.5142	0.5194	N/A	0.4484	0.4960	N/A
MSE (mean)	326.80	323.40	N/A	371.17	338.95	N/A
RMSE	N/A	N/A	18.59	N/A	N/A	18.94
R2 Score	N/A	N/A	0.5071	N/A	N/A	0.4886
MSE	N/A	N/A	345.76	N/A	N/A	358.69
MAE	N/A	N/A	13.09	N/A	N/A	13.43

7. CONCLUSION

- Data Preprocessing: Missing values were imputed using the mean, and outliers were removed using the IQR method. The data preprocessing steps are crucial for ensuring data quality before modeling.
- Feature Engineering: Several features, including time-based and interaction terms, were engineered. Feature engineering plays a significant role in improving model performance.
- Feature Selection: Permutation Importance with Random Forest was used to select the most important features. The 'energy_rolling_24' feature was identified as the most important predictor of energy consumption.
- Modeling: Both Random Forest and Gradient Boosting models were trained and evaluated. The Random Forest model slightly outperformed the Gradient Boosting model.

- Model Performance: The best-performing model, Random Forest, achieved an R2 score of 0.5071 and an RMSE of 18.59 on the test set. This indicates that the model explains approximately 50% of the variance in energy consumption.

8. CHALLENGES

- **Outlier Handling**:
  - Challenge: Outliers in the data can skew analysis and modelling results.
  - Solution: Outliers were removed to clean the data.
- **Feature Selection:**
  - Challenge: Determining the most relevant features for prediction.
  - Solution: Feature importance was determined using a Random Forest model.
- **Model Performance**:
  - Challenge: Achieving high accuracy in predicting equipment energy consumption.
  - Solution: Use of Random Forest and Gradient Boosting models.
- **Data Complexity**:
  - Challenge: The dataset includes many variables with complex relationships.
  - Solution: Feature selection and engineering to simplify the data.
- **Generalization:**
  - Challenge: Ensuring the model performs well on unseen data.
  - Solution: Model tuning and evaluation on test sets.

9. REFERENCES

https://www.mdpi.com/1996-1073/16/9/3748
https://www.datacamp.com/blog/machine-learning-projects-for-all-levels
https://www.sciencedirect.com/science/article/pii/S266616592030034X
https://www.geeksforgeeks.org/gradient-boosting-vs-random-forest/