Project Documentation

For

Power Pulse: Household Energy Usage Forecast

Version 1.0

Prepared by

Name: Mohamed Niyas

Batch No: MAE1

Release Notice

This is Version 1.0 of Power Pulse: Household Energy Usage Forecast Project Documentation

Name	Batch No	Organization
Prepared By		
Mohamed Niyas	MAE1	Guvi

Version History:

Version No.	Date Issued	Remarks
1.0	22 Feb 2025	Initial Release

Table of Contents

1 Introduction

PowerPulse is a machine learning project aimed at predicting household energy consumption based on historical data. The goal is to provide actionable insights into energy usage trends and deliver a predictive model that can help optimize energy consumption for households or serve as a baseline for further research into energy management systems.

1.1.1 Purpose Of the Document

This documentation will guide you through the following sections to help you set up, configure, and use PowerPulse project effectively:

- **Installation:** Step-by-step instructions to install all necessary dependencies and set up the project on your local machine.
- **Configuration:** Detailed guidance on configuring the application, including setting up the dataset.
- **Usage:** Instructions on how to run the application and navigate through its features.
- **Features:** An overview of the main features of Data Preprocessing, Feature Engineering, Regression Modelling, Evaluation Metrics
- **Data Analysis Options:** A list of the various data analysis options available within the application.
- **Insights and Challenges:** Insights gained during the development process and challenges faced, along with solutions implemented.
- **Troubleshooting:** Common issues and their solutions to help you resolve any problems you might encounter.

1.1.2 Scope

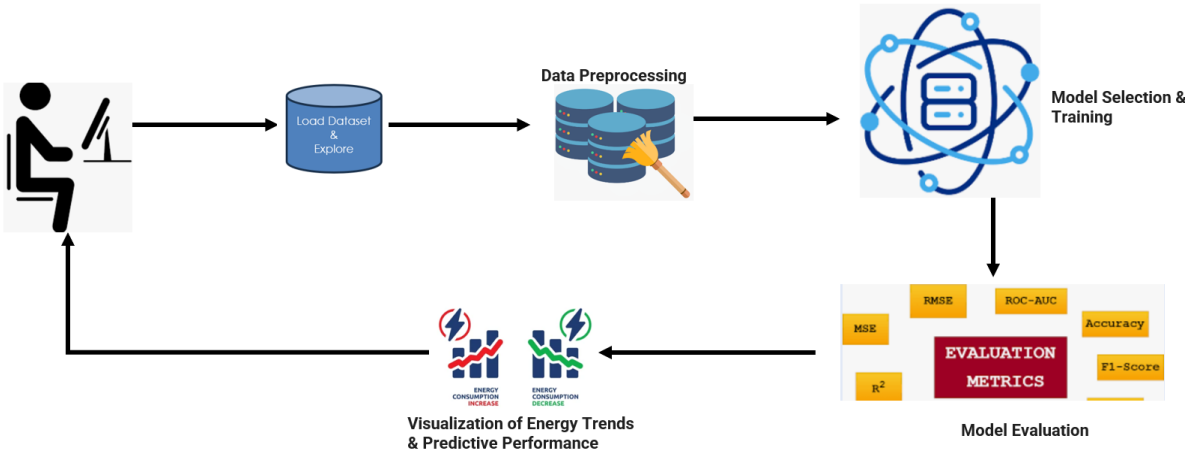
In the modern world, energy management is a critical issue for both households and energy providers. Predicting energy consumption accurately enables better planning, cost reduction, and optimization of resources.

1.1.3 Business Use cases

- Energy Management for Households: Monitor energy usage, reduce bills, and promote energy-efficient habits.
- Demand Forecasting for Energy Providers: Predict demand for better load management and pricing strategies.
- Anomaly Detection: Identify irregular patterns indicating faults or unauthorized usage.
- Smart Grid Integration: Enable predictive analytics for real-time energy optimization.

2 Execution Flow Diagram

2.1.1 Overview



3 Installation

3.1.1 Prerequisites

- Google Colab, or "Colaboratory," is a free, cloud-based platform that allows you to write and execute Python code in your browser.
- Dataset - Individual Household Electric Power Consumption

4 Clone the Repository

https://github.com/niyasatsg/PowerPulse-Household-Energy-Usage-Forecast

5 Install Dependencies

5.1.1 requirements.txt file

Create a requirements.txt file [Refer to the file from the above link]

5.1.2 install the dependencies:

pip install -r requirements.txt

6 Configuration

Dataset: Measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available. This archive contains 2075259 measurements gathered in a house located in Sceaux (7km of Paris, France) between December 2006 and November 2010 (47 months).

Variable Name	Role	Type	Description	Units	Missing Values
Date	Feature	Date			no
Time	Feature	Categorical			no
Global_active_power	Feature	Continuous			no
Global_reactive_power	Feature	Continuous			no
Voltage	Feature	Continuous			no
Global_intensity	Feature	Continuous			no
Sub_metering_1	Feature	Continuous			no
Sub_metering_2	Feature	Continuous			no
Sub_metering_3	Feature	Continuous			no

6.1.1 Source of the Dataset

https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption

7 Run the Application

household_energy_usage_forecast.py

7.1.1 Access the Application

- Open your web browser.
- Go to Google Colab
- Import provided .py file and run sections.
- Note: Random Forest and Gradient Boosting takes 5 to 7 minutes to complete the training

8 Project Report

8.1.1 Explore the Dataset

- Explore the given dataset before pre-processing.

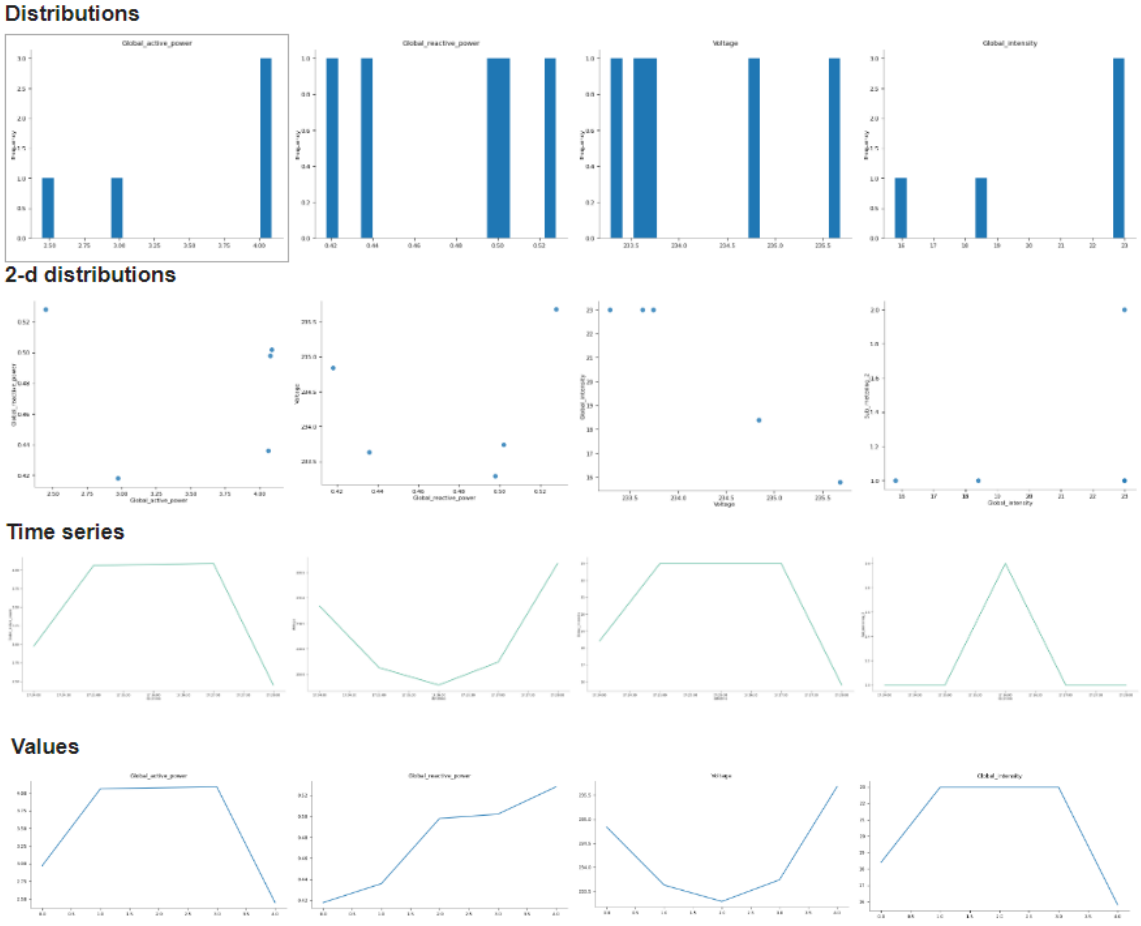
```
RangeIndex: 2075259 entries, 0 to 2075258
Data columns (total 8 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Global_active_power    float64
 1   Global_reactive_power  float64
 2   Voltage                float64
 3   Global_intensity       float64
 4   Sub_metering_1         float64
 5   Sub_metering_2         float64
 6   Sub_metering_3         float64
 7   datetime               datetime64[ns]
dtypes: datetime64[ns](1), float64(7)
memory usage: 126.7 MB
```

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	datetime
count	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2075259
mean	1.091615e+00	1.237145e-01	2.408399e+02	4.627759e+00	1.121923e+00	1.298520e+00	6.458447e+00	2008-12-06 07:12:59.999994112
min	7.600000e-02	0.000000e+00	2.232000e+02	2.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	2006-12-16 17:24:00
25%	3.080000e-01	4.800000e-02	2.389900e+02	1.400000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2007-12-12 00:18:30
50%	6.020000e-01	1.000000e-01	2.410100e+02	2.600000e+00	0.000000e+00	0.000000e+00	1.000000e+00	2008-12-06 07:13:00
75%	1.528000e+00	1.940000e-01	2.428900e+02	6.400000e+00	0.000000e+00	1.000000e+00	1.700000e+01	2009-12-01 14:07:30
max	1.112200e+01	1.390000e+00	2.541500e+02	4.840000e+01	8.800000e+01	8.000000e+01	3.100000e+01	2010-11-26 21:02:00
std	1.057294e+00	1.127220e-01	3.239987e+00	4.444396e+00	6.153031e+00	5.822026e+00	8.437154e+00	NaN

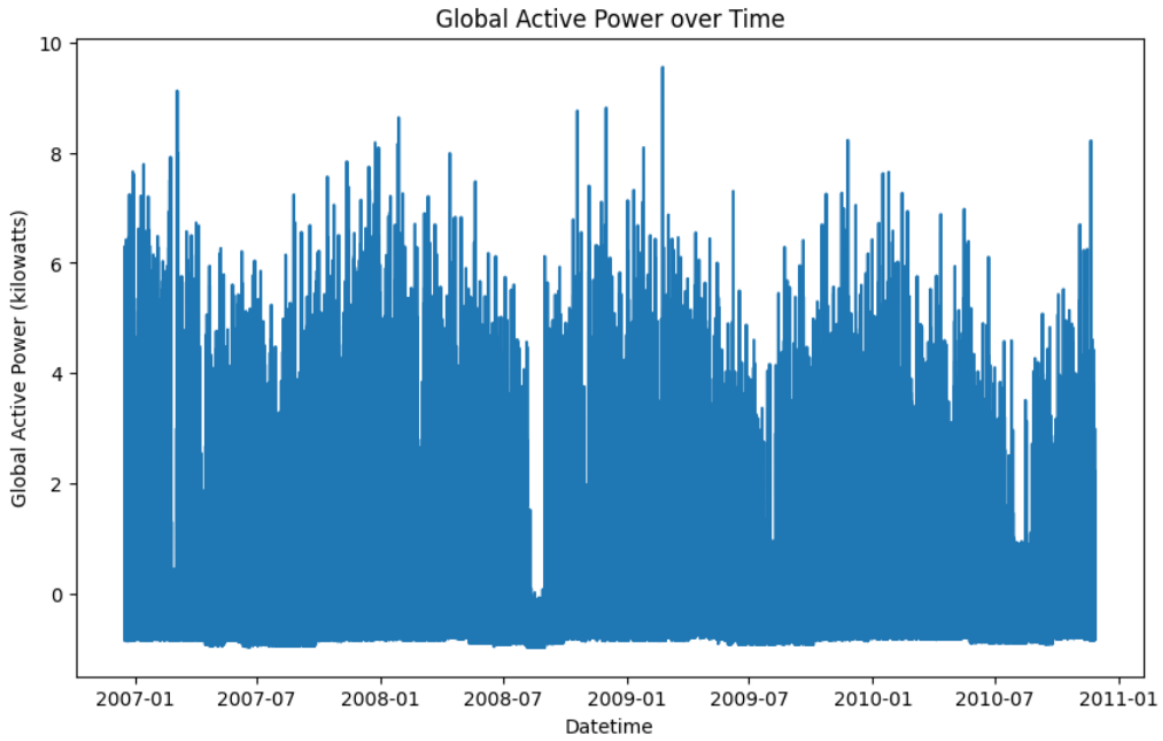
8.1.2 Data Preprocessing / EDA

- Handle Missing Values
- Parse Date and Time into Separate Features
- Create Additional Features
- Normalize/Scale Data

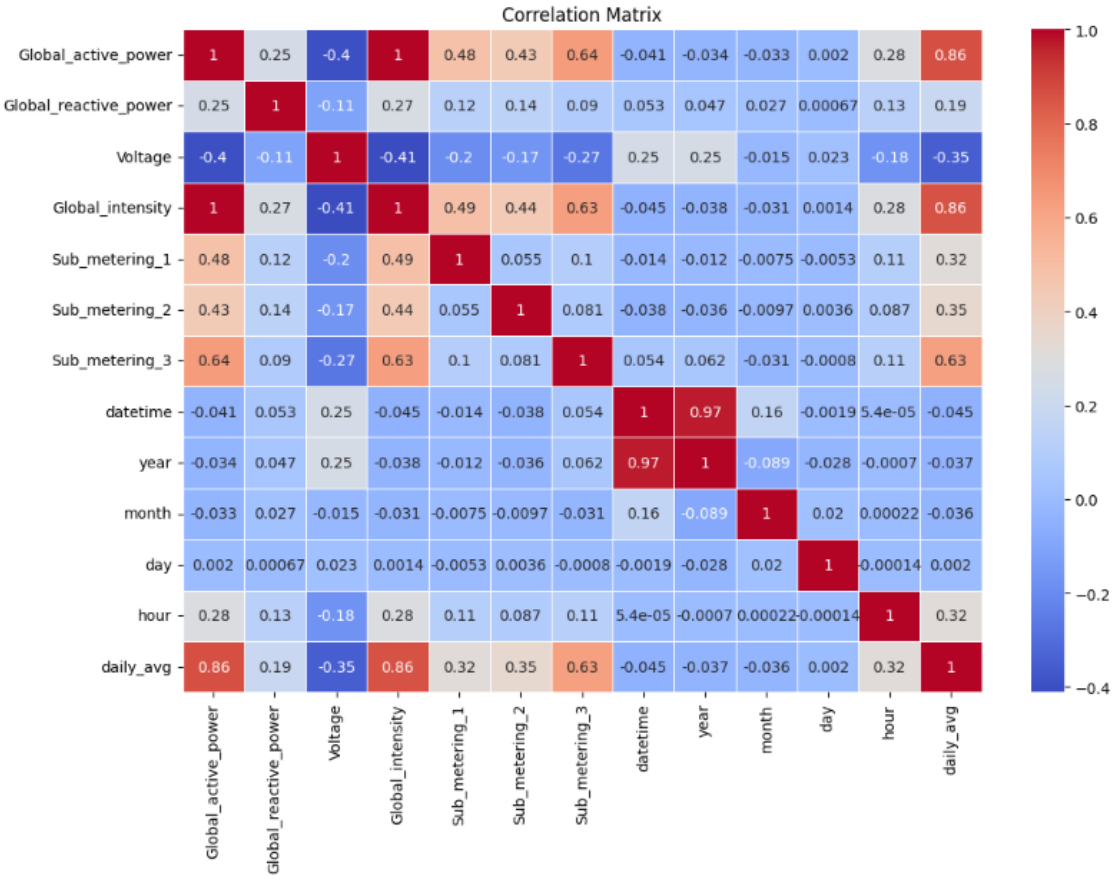
	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	datetime	year	month	day	hour
0	2.973749	0.418	234.84	18.4	0.0	1.0	17.0	2006-12-16 17:24:00	2006	12	16	17
1	4.062593	0.436	233.63	23.0	0.0	1.0	16.0	2006-12-16 17:25:00	2006	12	16	17
2	4.075918	0.498	233.29	23.0	0.0	2.0	17.0	2006-12-16 17:26:00	2006	12	16	17
3	4.089243	0.502	233.74	23.0	0.0	1.0	17.0	2006-12-16 17:27:00	2006	12	16	17
4	2.450266	0.528	235.68	15.8	0.0	1.0	17.0	2006-12-16 17:28:00	2006	12	16	17



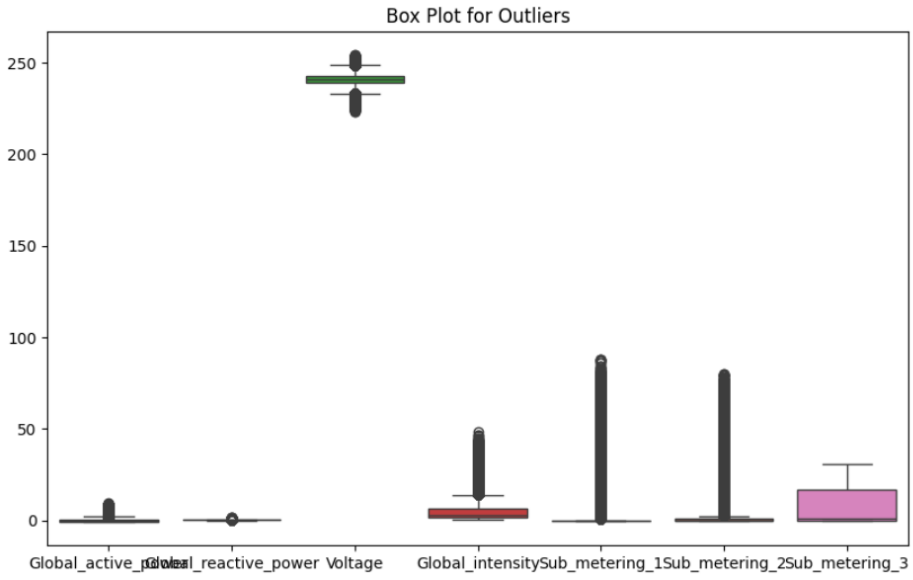
- Time series plot



- Compute the correlation matrix



- To identify outliers



8.1.3 Model Selection and Training

- Impute missing values
- Split the dataset

8.1.4 Train Regression Models: Linear Regression

```
LinearRegression
LinearRegression()
```

8.1.5 Train Regression Models: RandomForest

- Use a Smaller Subset of Data
- Reduce the Number of Trees

```
RandomForestRegressor
RandomForestRegressor(n_estimators=50)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=10)
```

- Dask for Parallel Computing

8.1.6 Train Regression Models: Gradient Boosting

```
GradientBoostingRegressor
GradientBoostingRegressor()
```

8.1.7 Train Regression Models: *HistGradientBoosting*

```
HistGradientBoostingRegressor
HistGradientBoostingRegressor()
```

8.1.8 Train Regression Models: *XGBoost*

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

8.1.9 *Evaluate Models:*

Linear Regression	RMSE: 0.5024912738439729, MAE: 0.27049345684476706, R2: 0.7476186672051445
Random Forest	RMSE: 0.49461719419544664, MAE: 0.2672750434333968, R2: 0.7554663671392862
Gradient Boosting	RMSE: 0.49642112738049327, MAE: 0.2695451680412813, R2: 0.7536794225865905
HistGradientBoosting	RMSE: 0.49346743357144557, MAE: 0.26743703652279915, R2: 0.7566019053555643
XGBoost	RMSE: 0.49472564350146986, MAE: 0.2680582851857959, R2: 0.7553591229495771

- **Best-Performing Model** Based on the evaluation metrics, the best-performing model is ***HistGradientBoosting*** with the following scores:

  RMSE: 0.4933 MAE: 0.2674 R^2: 0.7567

- ***How the Best-Performing Model is Identified***: The best-performing model is identified by comparing the evaluation metrics:

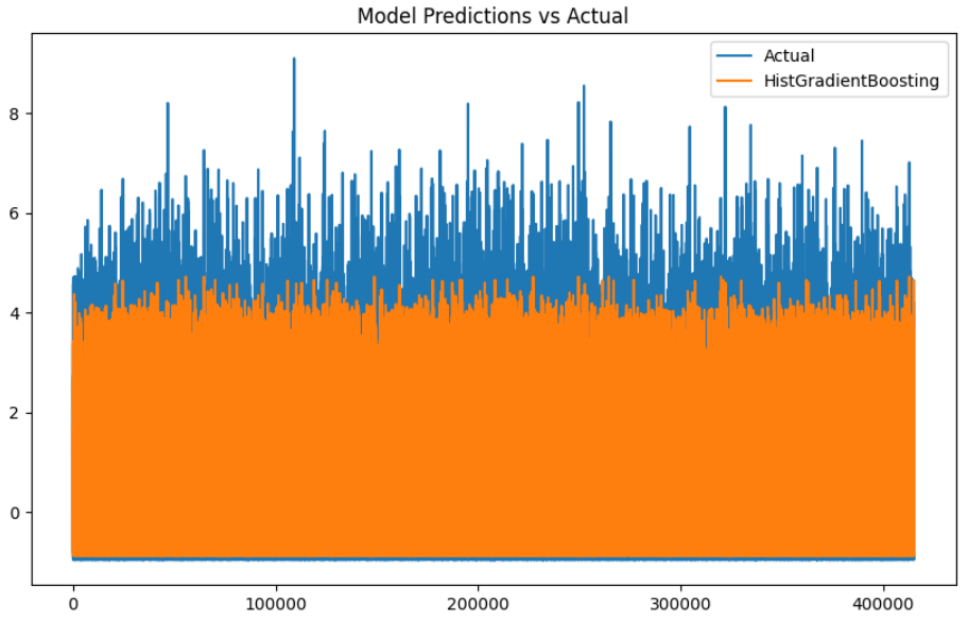
  - *Root Mean Squared Error (RMSE):* Measures prediction accuracy. Lower RMSE indicates better accuracy.

- *Mean Absolute Error (MAE):* Evaluates average error magnitude. Lower MAE indicates better performance.
- *R-Squared (R^2):* Indicates how well the model explains the variability of the target variable. Higher R^2 indicates better explanatory power. The HistGradientBoosting model has the lowest RMSE and MAE, and the highest R^2, making it the best-performing model among the ones evaluated.

8.1.10 *Visualization of Energy Trends and Predictive Performance*

Effective visualizations can help in understanding energy trends and the performance of the predictive model. Here are some suggested visualizations:

- **Model Predictions vs Actual**: Compare the model's predictions with actual values.
- These visualizations will help you present your findings effectively and provide actionable insights into energy usage trends



8.1.11 Summarization:

1. Accurate Prediction Model for Household Power Consumption
   The *HistGradientBoosting* model was identified as the best-performing model with the following metrics:

   RMSE: 0.4934 MAE: 0.2673 R^2: 0.7567 This model provides accurate predictions of household power consumption, which can be used for better energy management and planning.

2. Clear Insights into Key Factors Influencing Energy Usage Feature importance analysis helps in understanding which factors most influence energy usage. For example, features like hour, day, month, and daily_avg can provide insights into usage patterns:

Hour: Identifies peak usage times during the day. Day: Highlights daily consumption trends. Month: Shows seasonal variations in energy usage. Daily Average: Provides a smoothed view of consumption over time.

9 Insights and Challenges

9.1.1 Insights

1. **Best-Performing Model**:
    - The **HistGradientBoosting** model was identified as the best-performing model with the lowest RMSE (0.4934), lowest MAE (0.2674), and highest R^2 (0.7567). This model provided the most accurate predictions for household energy consumption.
2. **Feature Importance**:
    - Key features influencing energy usage included hour, day, month, and daily_avg. These features helped in understanding usage patterns and trends.
3. **Model Evaluation Metrics**:
    - **Root Mean Squared Error (RMSE)**: Measures prediction accuracy. Lower RMSE indicates better accuracy.
    - **Mean Absolute Error (MAE)**: Evaluates average error magnitude. Lower MAE indicates better performance.
    - **R-Squared (R^2)**: Indicates how well the model explains the variability of the target variable. Higher R^2 indicates better explanatory power.
4. **Visualization**:
    - Effective visualizations, such as time series plots, correlation matrices, and feature importance plots, provided clear insights into energy trends and model performance.

9.1.2 Challenges

- ***Handling Large Datasets:*** The dataset size (132,960,755 bytes) posed challenges in terms of memory usage and processing time. Techniques like chunking, optimizing data types, and using Dask for parallel computing were employed to manage this.
- ***Missing Values:*** Handling missing values was crucial for accurate model training. Imputation strategies, such as filling missing values with the mean, were used to address this issue.
- ***Crash when Running Random Forest:*** Random Forest model might be struggling with the large data
- ***Model Training and Evaluation:*** Training complex models like Random Forest and Gradient Boosting required significant computational resources.

Upgrading to Google Colab Pro or using smaller subsets of data helped mitigate this.

- ***Hyperparameter Tuning:*** Hyperparameter tuning for models like Random Forest and Gradient Boosting was computationally intensive. Using RandomizedSearchCV instead of GridSearchCV helped reduce the computational load.

10 Troubleshooting

10.1.1 Common Issues

- **Memory Errors**:

  Running out of memory during model training and hyperparameter tuning was a common issue. Solutions included using smaller data subsets, reducing the number of trees in Random Forest, and upgrading to Google Colab Pro.

- **File Not Found Errors**:

  Ensuring the correct file path and uploading the dataset to Google Colab were necessary to avoid FileNotFoundError.

- **Model Evaluation Errors**:

  Ensuring compatibility with the version of scikit-learn used was important to avoid errors like TypeError when calculating RMSE.