Energy Forecasting Using Machine Learning

Ismail Al-Jawabrah

Objective

The goal of this project is to forecast Global Active Power consumption using historical household energy usage data.

By leveraging time-series analysis and machine learning models, we aim to provide accurate predictions that support energy planning, optimization, and resource management.

This task demonstrates the use of regression models and feature engineering techniques to improve forecasting performance and interpretability.

Data Description

The dataset used in this project contains minute-level measurements of electrical power consumption for a single household over several days. It includes both raw sensor readings and engineered features to enhance forecasting accuracy.

Original Features:

- **Global Active Power:** Household energy usage in kilowatts.
- **Global Intensity:** Current intensity in amperes.
- **Voltage:** Line voltage in volts.
- **Sub_metering_1, 2, 3:** Energy consumption by different household zones.

Engineered Features:

- **Time-based features:** day, month, day of week, hour, and is_weekend.
- **Lag features:** 1-day, 2-day, and 7-day lags for key variables.
- **Rolling statistics:** 1-day rolling mean and standard deviation to capture temporal trends.

Cleaning Process:

- Missing values resulting from lag and rolling operations were dropped.
- Final dataset size after cleaning: **1391 rows × 15 features**.
- These enhancements aimed to enrich the temporal context of the data and improve model learning capability.

Preprocessing Summary

To prepare the dataset for modeling, several preprocessing steps were applied to ensure data quality and relevance:

- **Datetime Indexing:**
  - ➢ Converted the timestamp column to datetime format and set it as the DataFrame index to support time-series operations.

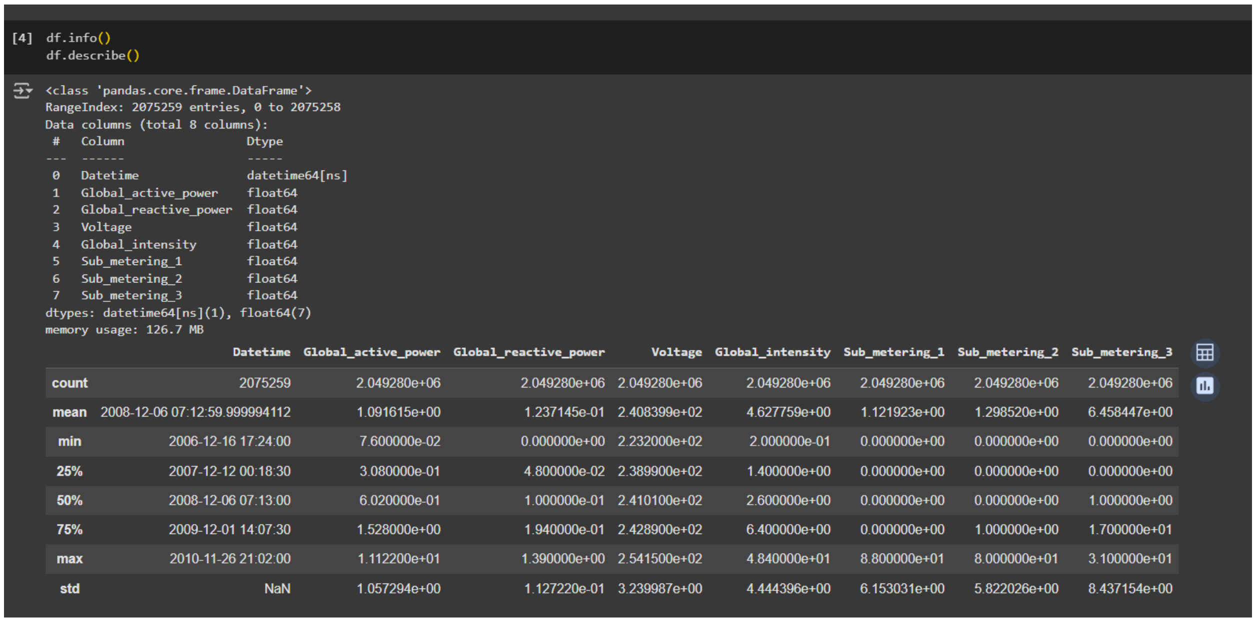
- **Missing Values Handling:**
  - ➢ Missing values introduced by lag and rolling operations were dropped to avoid misleading patterns.
  - ➢ No imputation was done to preserve temporal integrity.

- **Data Type Conversion:**
  - ➢ Ensured all numerical features were properly cast to float or int where applicable.

- **Feature Selection:**
  - ➢ Removed redundant columns (such as exact timestamps if not used).
  - ➢ Retained only relevant variables that contributed to modeling performance.

- **Final Shape:**
  - ➢ The cleaned and preprocessed dataset consisted of 1391 rows and 15 features ready for modeling.

- **Sample Overview Before Preprocessing:**
    - ➢ The figure below shows an initial snapshot of the dataset, including datatypes and basic descriptive statistics.



```
[4] df.info()
    df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075259 entries, 0 to 2075258
Data columns (total 8 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Datetime               datetime64[ns]
 1   Global_active_power    float64
 2   Global_reactive_power  float64
 3   Voltage                float64
 4   Global_intensity       float64
 5   Sub_metering_1         float64
 6   Sub_metering_2         float64
 7   Sub_metering_3         float64
dtypes: datetime64[ns](1), float64(7)
memory usage: 126.7 MB
```

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

These preprocessing steps helped improve the quality of the input data and ensured better performance for time-series forecasting models.

Time Series Decomposition and Autocorrelation Analysis

To better understand the temporal patterns in the energy consumption data, I performed seasonal decomposition using the seasonal_decompose function from statsmodels. This revealed clear trend, seasonality, and residual components in the time series. The seasonal component showed a recurring daily pattern in power usage.

Additionally, I plotted the Autocorrelation Function (ACF) to examine the dependency structure across different time lags. The ACF plot highlighted strong autocorrelation at daily intervals (e.g., lag=1440 for minute-level data), indicating significant temporal relationships in the dataset.

These insights directly informed the selection of lag features and rolling statistics during feature engineering and supported the assumption of temporal dependencies crucial for effective time-series forecasting.

Modeling and Evaluation

To evaluate the performance of different regression algorithms on the time-series dataset, three models were trained and tested:

- **Linear Regression:**
    - The Linear Regression model was trained using the training dataset and evaluated on the test set.
    - Performance metrics were computed, and predictions were visualized against the actual values.

- **MSE:** 0.0000
- **RMSE:** 0.0000
- **MAE:** 0.0000
- **R² Score:** 1.0

    - This perfect performance suggests either an ideal linear relationship or potential data leakage, which must be reviewed.

- **Random Forest Regressor:**
    - Random Forest, a tree-based ensemble model, was also trained and evaluated:

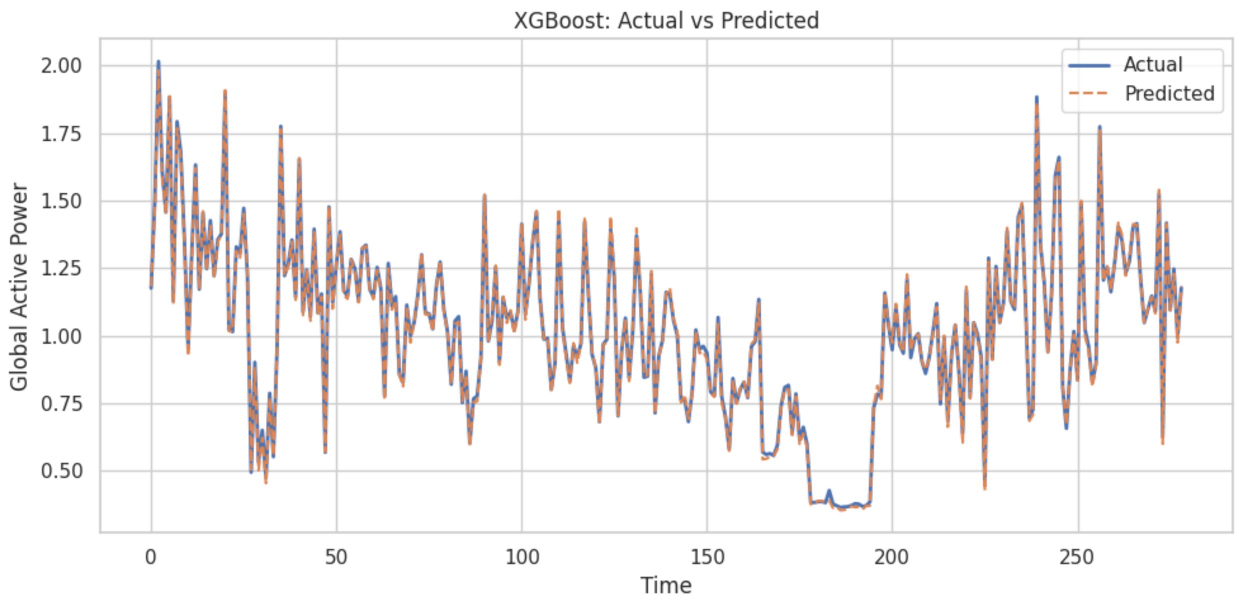
- **MSE:** 0.00022
- **RMSE:** 0.01480
- **MAE:** 0.01171
- **R² Score:** 0.99790

    - This model showed excellent performance, closely matching the actual values and generalizing well.

- **XGBoost Regressor:**
  - ➢ The third model was XGBoost, a gradient boosting framework optimized for performance:

- **MSE:** 0.00017
- **RMSE:** 0.0130
- **MAE: 0**.01024
- **R² Score:** 0.99838

  - ➢ XGBoost achieved the best results among all models, balancing accuracy and generalization.



Summary of Evaluation:

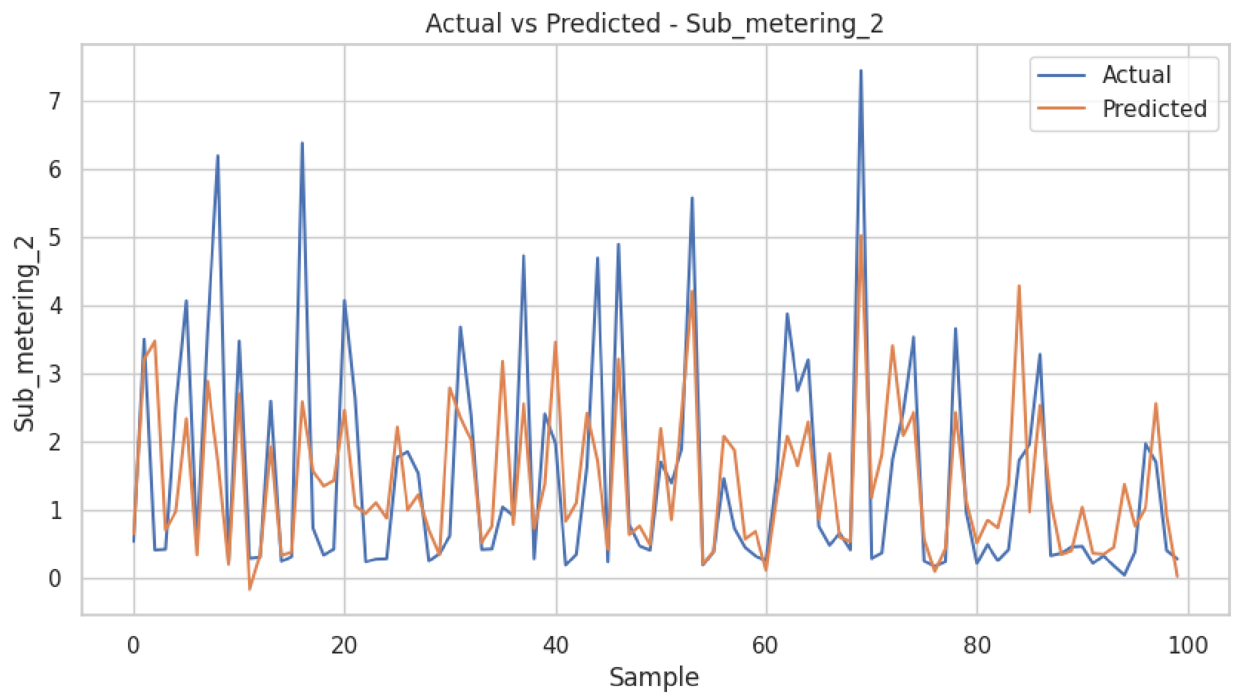
Among the three regression models tested, **XGBoost Regressor** achieved the best overall performance, offering a strong balance between accuracy and generalization with an R² score of 0.9984. While **Linear Regression** scored a perfect R² of 1.0, this result may indicate overfitting or data leakage and requires further inspection. The **Random Forest Regressor** also performed excellently with high accuracy and reliable predictions.

Based on the results, **XGBoost** was selected for final deployment and visualization.

External Weather Data Integration

As an extension, weather data was integrated into the original dataset to test whether temperature and humidity affect energy consumption.

- Weather data was collected using Open-Meteo API.
- Features added: temperature_2m_max, relative_humidity_2m_mean.
- Models were retrained using the updated dataset.
- Best Model (**XGBoost**):
  - MSE: 1.15
  - MAE: 0.74
  - R^2 Score: 0.41



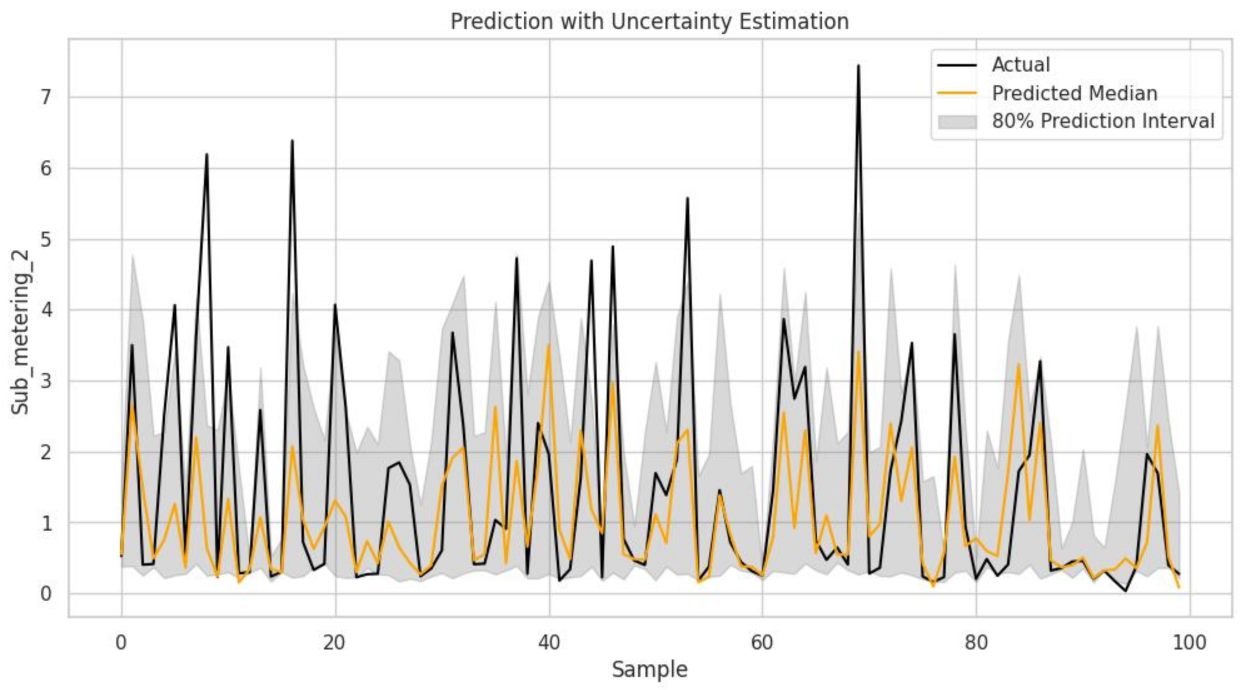
While the performance dropped, this experiment showed that external weather data has some predictive value, though limited in this case.

Uncertainty Estimation

To estimate the prediction uncertainty, quantile regression was implemented using **GradientBoostingRegressor** for:

- 10th percentile (lower bound)
- 50th percentile (median prediction)
- 90th percentile (upper bound)

A visual plot was created showing the prediction interval (shaded area) between the 10th and 90th percentiles.



This helps understand the confidence range of our predictions, which is important for real-world applications such as planning or risk analysis.