ENERGY CONSUMPTION AND PREDICTION PROJECT

..

Problem statement

- Develop a predictive model to forecast the energy consumption of households based on historical power consumption data, conditions, and demographic information. The goal is to identify trends and help users optimize their energy usage

Data Importing

- Reading Dataset
- The file 'hpc.txt' is read with a semicolon (;) separator
- hpc = household_power_consumption
- Exploring the data by using head(),tail(),describe(),shape,info() to get idea about how the data is

Data Cleaning

Convert Date and Time to Datetime Format

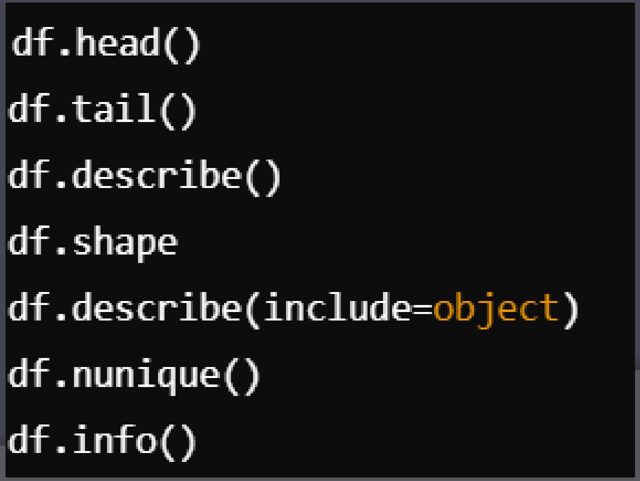
- Combined 'Date' and 'Time' columns to create a new 'Datetime' column
- I combined date and time colums because Use time-based indexing to easily filter or retrieve data from a specific time range and Calculate time intervals between different records

Handled missing values:

The dataset consists of multiple columns, including both numerical and object

- To get accurate Null values , first we have to convert columns which are in string to numericals
- Columns that represent numerical values but stored as strings are converted using 'to_numeric'

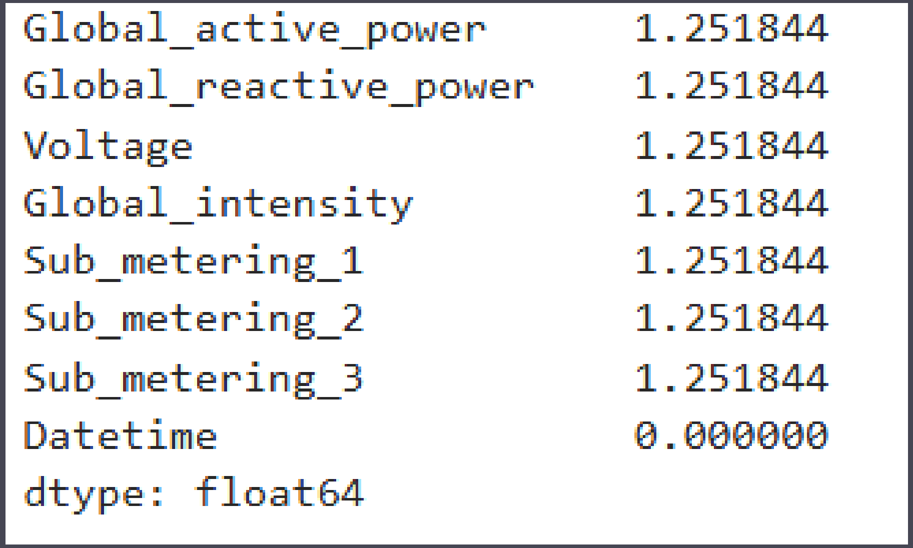
- Feature Engineering:Created holiday and sunlight indicators.
  - Convert the tuple to a date object
  - Converting tuple (year, month, day) to date object
  - Apply the function to the 'Date' column (which is in (year, month, day) format)
  - Assuming 'time' is a tuple (hour, minute)
  - Sunlight is present between 06:00 and 18:00
  - Apply the function to the 'Time' column and create 'Sunlight' column
- Checking data structures



```
df.head()
df.tail()
df.describe()
df.shape
df.describe(include=object)
df.nunique()
df.info()
```

we can deal with null values by fill missing values with the mean , median of the column or delete row which have null values

i am filled the mean of the column



```
Global_active_power      1.251844
Global_reactive_power    1.251844
Voltage                  1.251844
Global_intensity         1.251844
Sub_metering_1           1.251844
Sub_metering_2           1.251844
Sub_metering_3           1.251844
Datetime                 0.000000
dtype: float64
```

the percentage of null values in each column

Mathematical Insight - Data Cleaning

- Missing values percentage:
  (Missing Count/Total Rows)×100
- Ensures unbiased results for linear models.
- Filled using

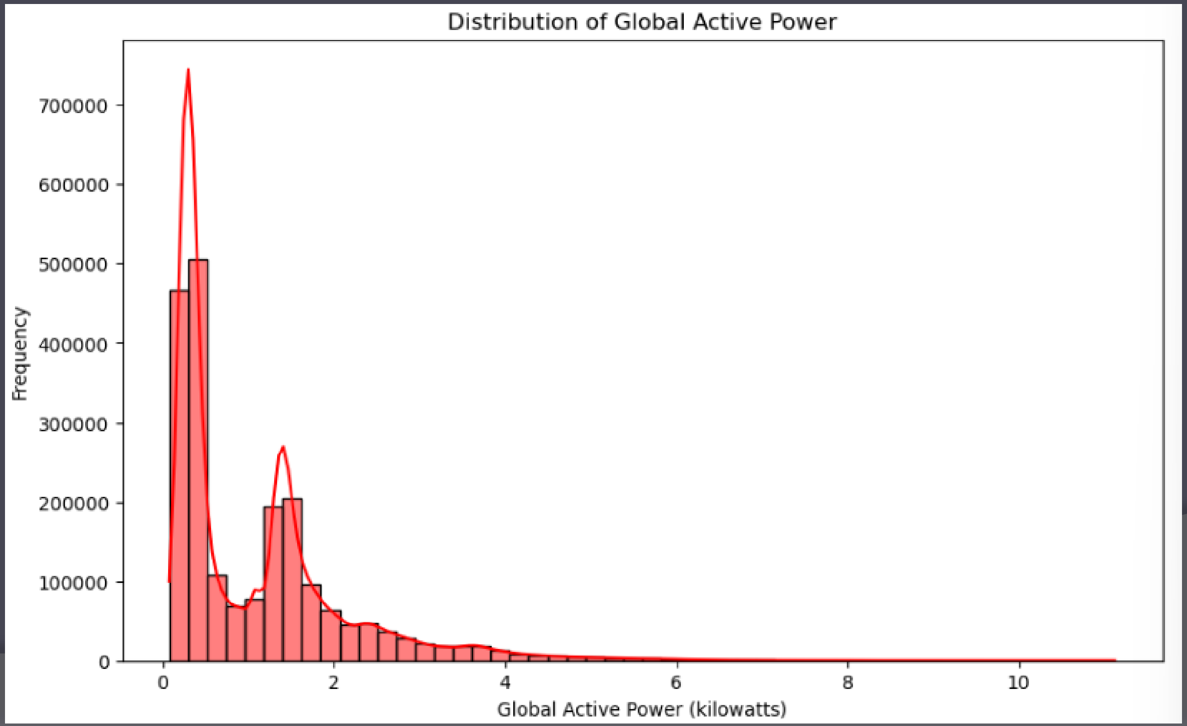
$$\text{Imputed Value (Mean)} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Data Visualization

What is the trend of 'Global_active_power' over time?

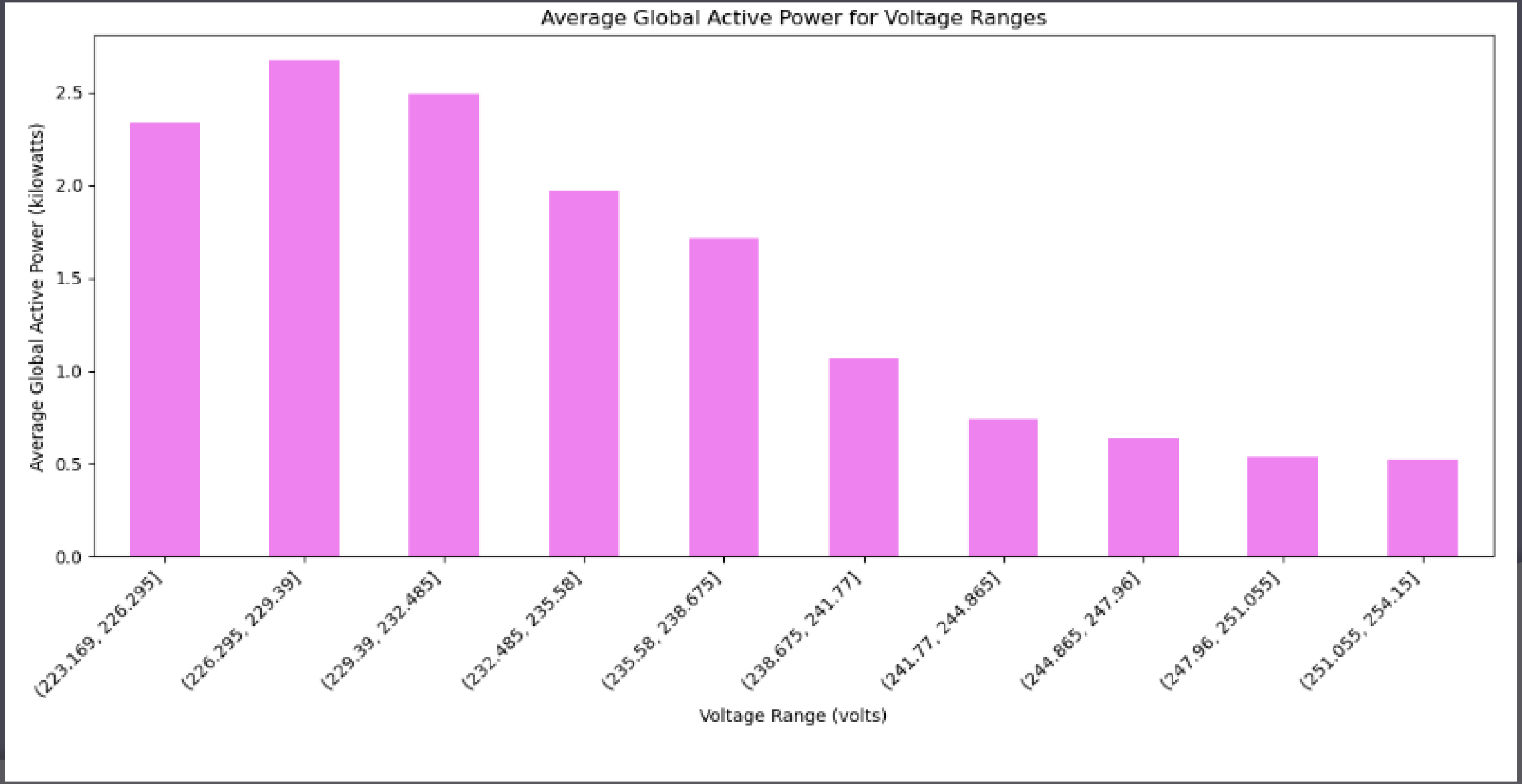
creates a histogram to show the distribution of "Global Active Power" in the dataset. It divides the data into 50 bins and adds

smooth curve on top to highlight the overall pattern. The plot is labeled with the title and axis descriptions, and it’s displayed on the screen.



What is the relationship between 'Global_active_power' and 'Voltage’?

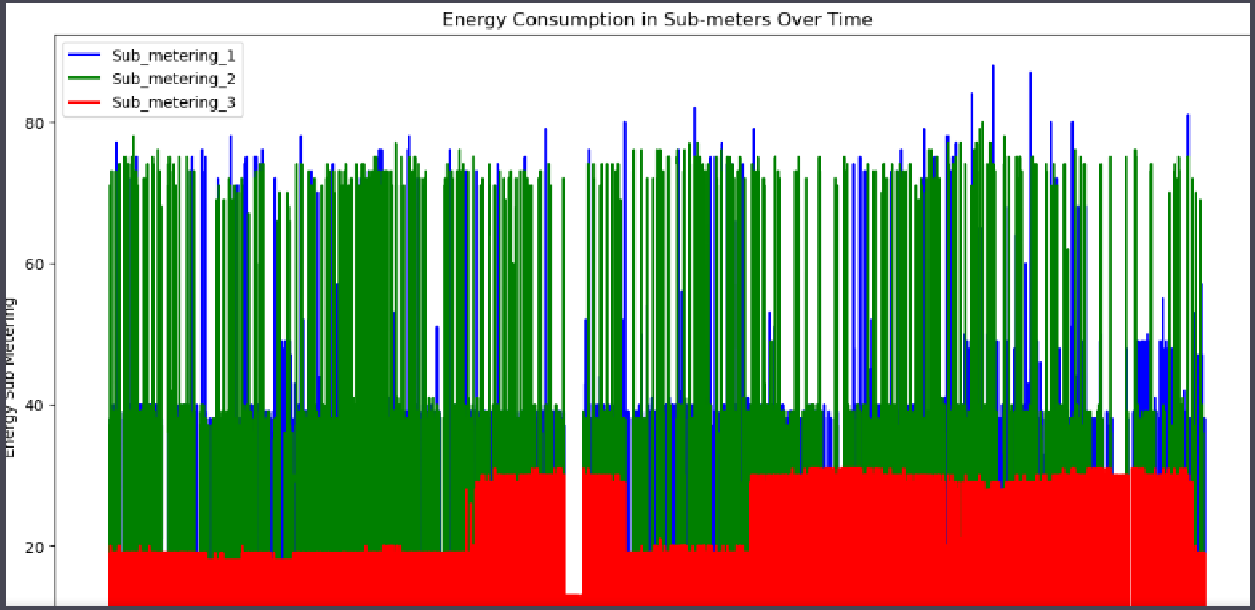
This groups the "Voltage" values into 10 equal ranges and calculates the average "Global Active Power" for each range. It then creates a barchart with the voltage ranges on the x-axis and the average power on the y-axis, displaying the relationship between voltage levels and power usage.



What is the daily average of 'Sub_metering' values?

This code creates a line plot to visualize the energy consumption from three sub-meters ('Sub_metering_1', 'Sub_metering_2', and 'Sub_metering_3') over time.

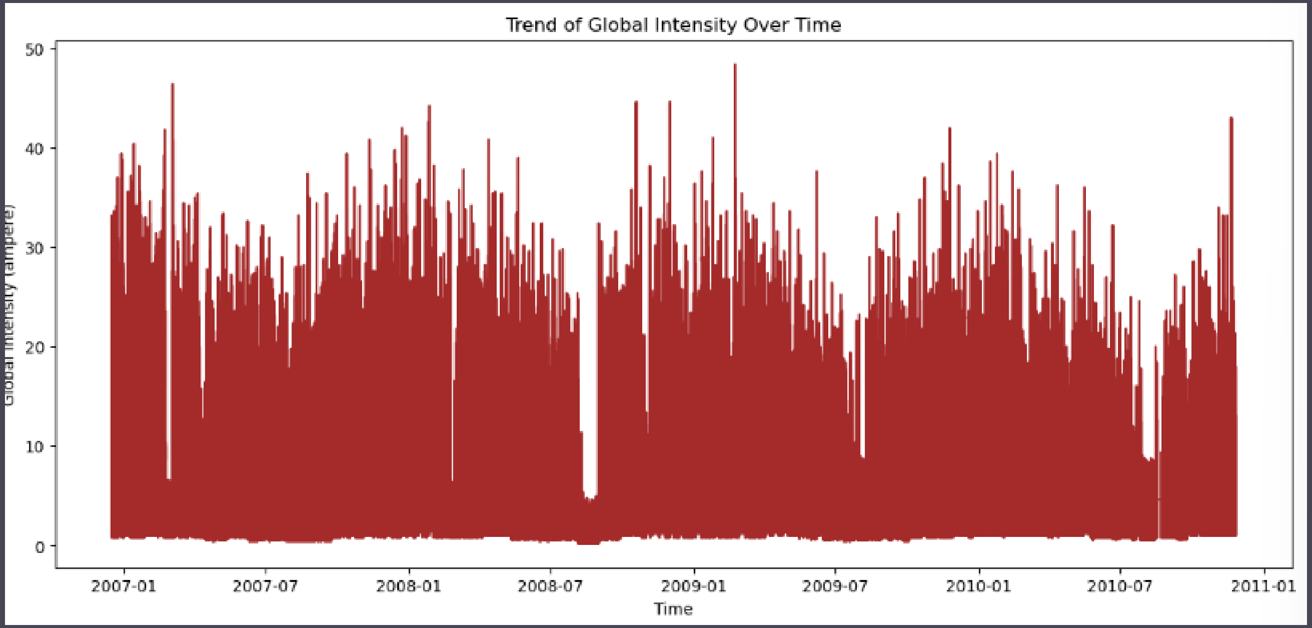
It plots each sub-meter's data against the Datetime column, with labels for each sub-meter and a legend to differentiate them



the energy consumption in the three sub-meters over time shows me clear trends and differences in how each sub-meter is used.

How does 'Global_intensity' vary over different hours of the day?

This code generates a line plot to display how "Global_intensity" changes over time. It uses the Datetime column for the x-axis and the "Global_intensity" for the y-axis, with the line colored purple.



Histogram: Distribution analysis through bins

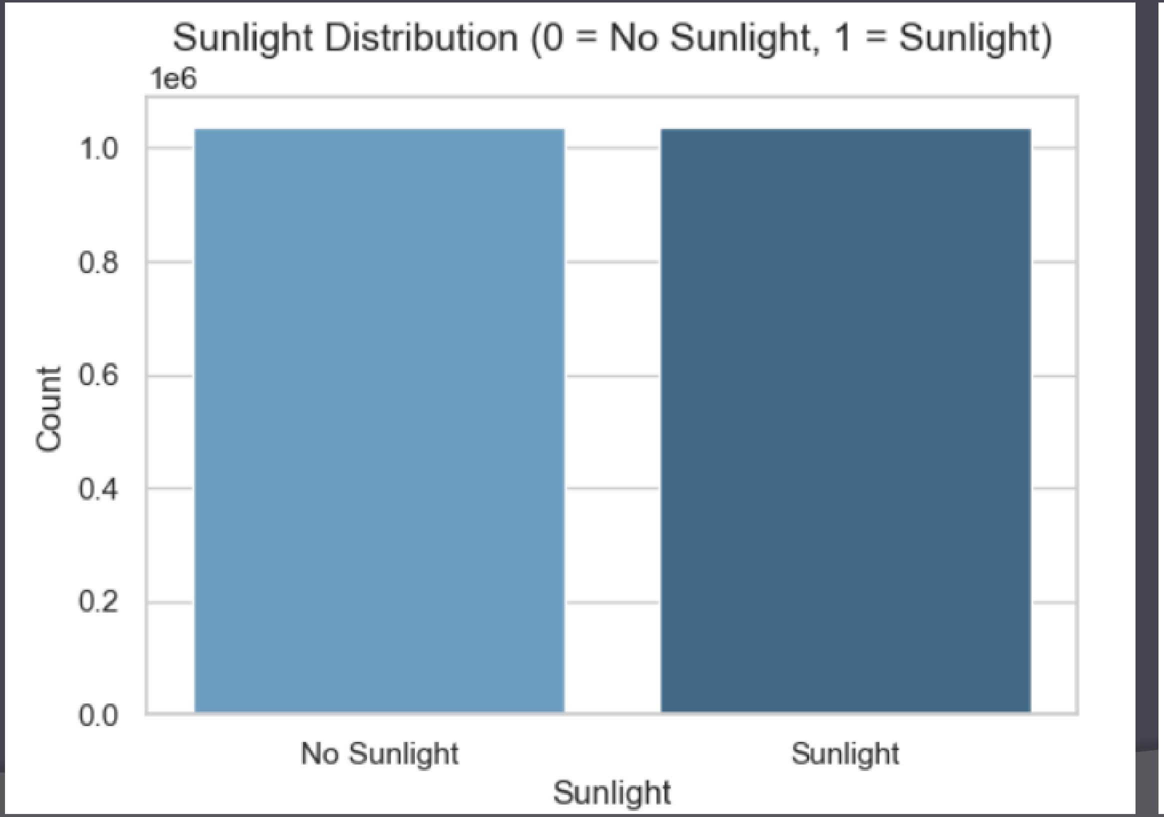
.

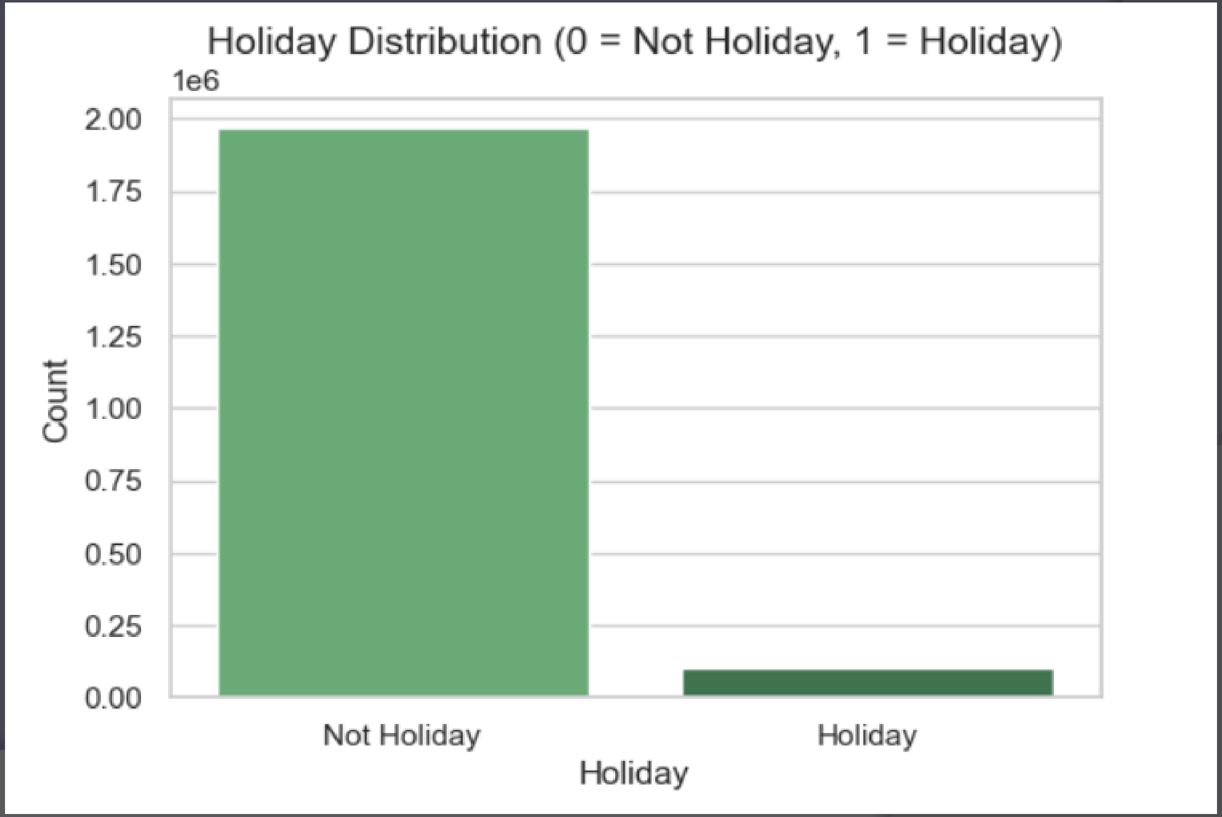
$$\text{Frequency Density} = \frac{\text{Frequency}}{\text{Bin Width}}$$

Bar charts for holiday and sunlight distributions

two bar plots: one for the distribution of sunlight (Sunlight column) and one for holidays (Holiday column).

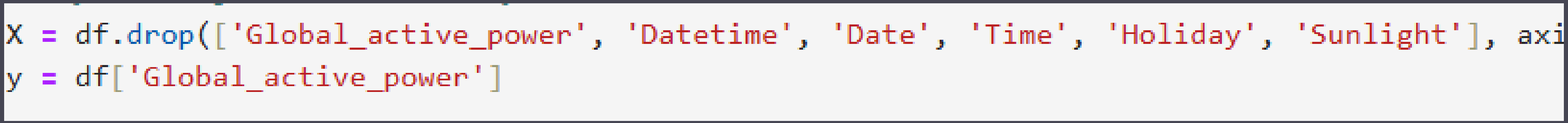
Each plot shows the count of occurrences for values 0 and 1, where 0 means "No Sunlight"/"Not Holiday," and 1 means "Sunlight"/"Holiday."





Model Building

- **Feature Selection:**
  Excluded time-based and categorical features, keeping only numerical variables for modeling.



```
X = df.drop(['Global_active_power', 'Datetime', 'Date', 'Time', 'Holiday', 'Sunlight'], axi
y = df['Global_active_power']
```

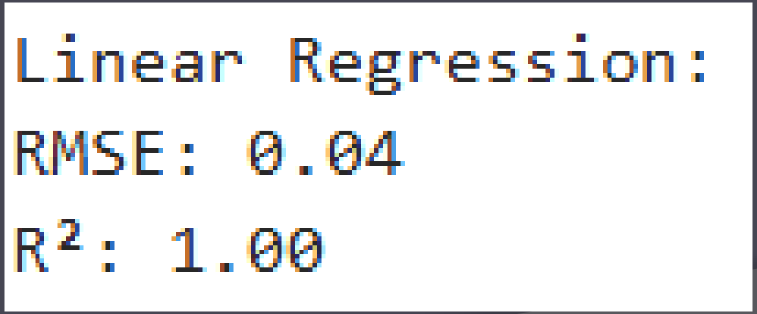
- **Train-Test Split:**
  Split the data into 80% training and 20% testing sets to evaluate model performance

Split the data into train and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Linear Regression

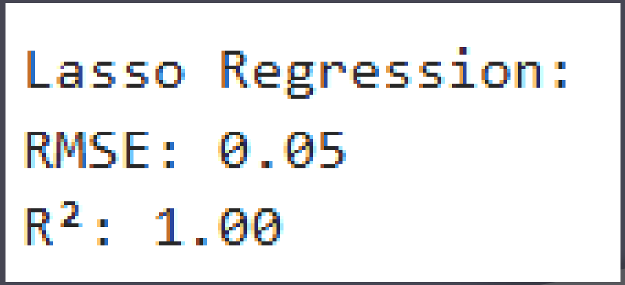
- Linear Regression is a basic machine learning model used to predict a continuous outcome (dependent variable, y) based on one or more independent variables (X). The goal is to find a linear relationship between X and y by fitting a straight line to the data.
- Model Initialization : LinearRegression() initialize the model
- Model Training:fir(x_train,y_train) learns the best fit by minimizing the error between actual and predicted values
- Prediction : predict(x_test) computes predictions using learned line
- Evalution : RMSE , R^2



```
Linear Regression:
RMSE: 0.04
R²: 1.00
```

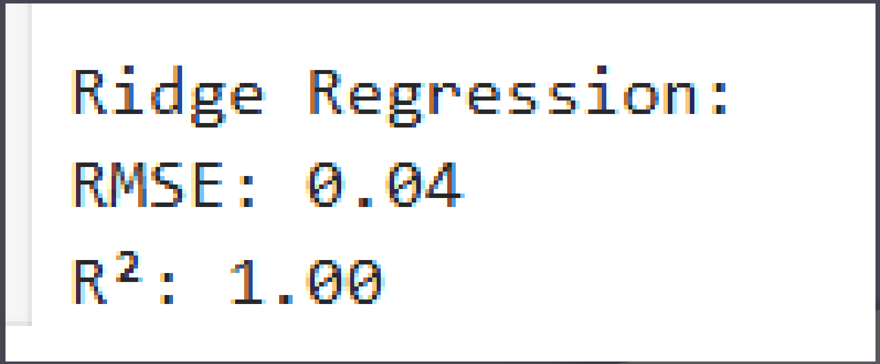
Lasso Regression

- Lasso (Least Absolute Shrinkage and Selection Operator) is a type of linear regression that adds a penalty for large coefficients to reduce overfitting and perform feature selection. The penalty term ensures that some coefficients may shrink to exactly zero, effectively removing less important features.
- Initialization: Lasso(alpha=0.1)adds regularization to reduce overfitting and select features.
- Fit:lasso_model_fit(x_train,y_train) trains the model
- Prediction:lasso_model_predict(x_test) makes predictions
- **Evaluation**: RMSE evaluates error; R^2 measures explained variance.


```
Lasso Regression:
RMSE: 0.05
R²: 1.00
```

Ridge Regression

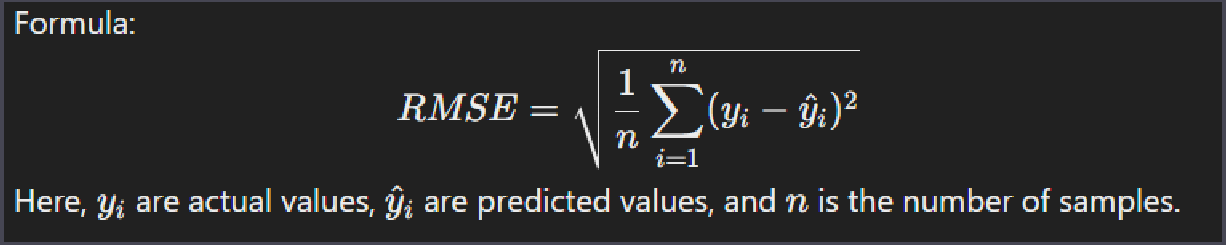
- **Ridge Regression** is a type of linear regression that includes a penalty term (regularization) to prevent overfitting by shrinking the regression coefficients. This is particularly useful when dealing with multicollinearity (correlated predictors) or high-dimensional data.
- When α=0, Ridge Regression behaves like standard linear regression. A higher α\alphaα leads to more shrinkage of coefficients, reducing model complexity.
- **RMSE**: Measures how well the model predictions match actual values (lower is better).
- **R²**: Proportion of variance in the dependent variable explained by the model (closer to 1 is better)



```
Ridge Regression:
RMSE: 0.04
R²: 1.00
```

Data and Model Analysis

- The purpose of this step is to assess the performance of the models used in predicting Global_active_power
- This involves evaluating how well each model explains the variation in the target variable and how accurately it predicts unseen data. Additionally, we compare models to identify the one that balances simplicity and effectiveness.
- Root Mean Square Error (RMSE):
- RMSE is used to measure the average magnitude of the prediction errors


Formula:

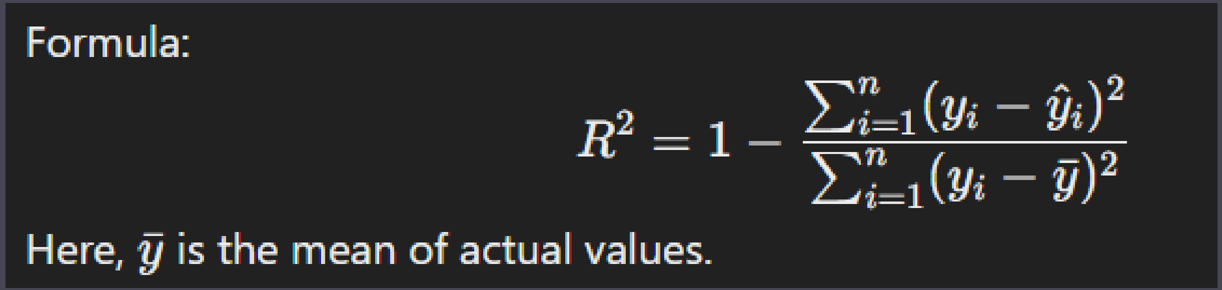
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Here, y_i are actual values, $\hat{y}_i$ are predicted values, and n is the number of samples.

- Lower RMSE indicates that model predictions are closer to actual values

R^2 Score (Coefficient of Determination):

- R^2 measures how much of the variance in the dependent variable is captured by the model



Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Here, $\bar{y}$ is the mean of actual values.

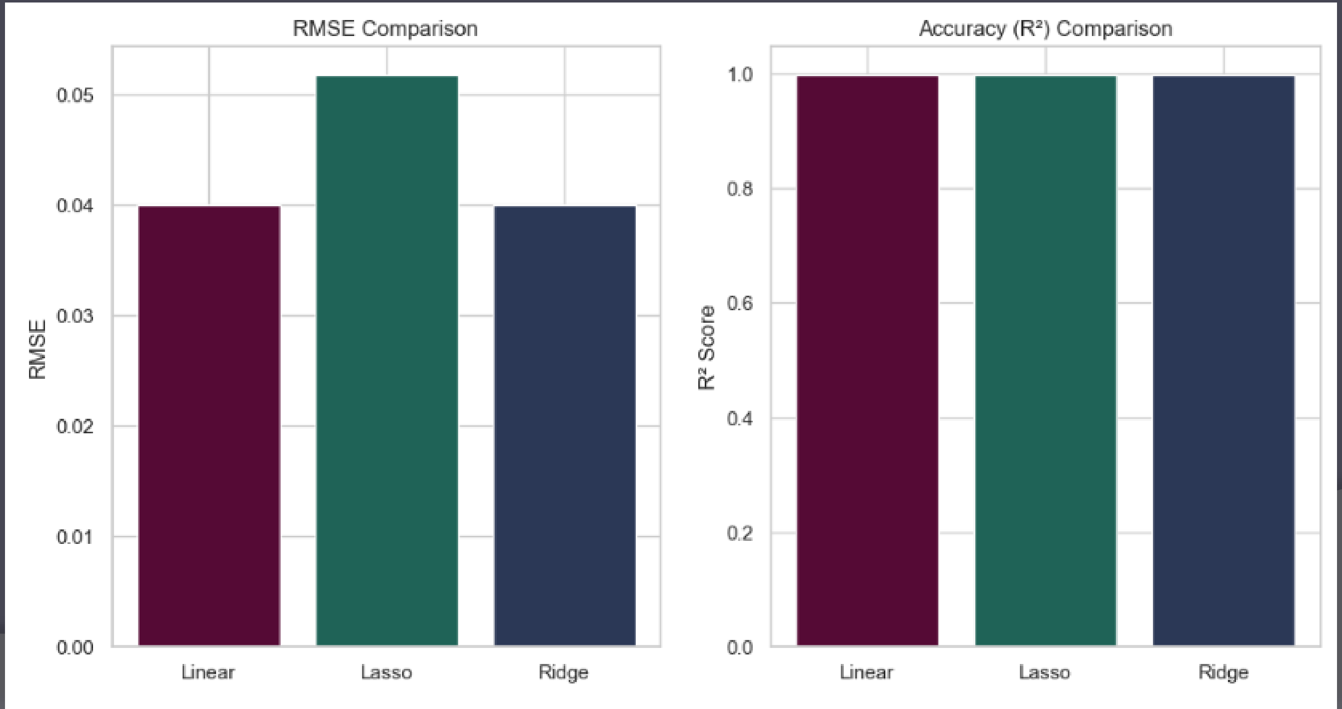
- R^2 ranges from 0 to 1:**1**: Perfect prediction.
- **0**: Model explains no variance.
- Negative values indicate the model performs worse than a simple average.

Observation:

In the RMSE plot, Ridge and Linear Regression show almost identical performance (RMSE: 0.04), indicating they fit the data well. Lasso has a slightly higher RMSE (0.05), likely due to its feature selection.

In the R² plot, all models have values close to 1.00, meaning they explain almost all the variance in the data. This suggests the data is well-suited for linear models.

Overall, Ridge and Linear Regression perform almost the same, while Lasso performs slightly worse due to its complexity reduction.



Why certain models have good accuracy and others don't

Linear Relationships in Data

- The strong linear relationships allow Linear, Ridge, and Lasso models to capture patterns effectively with minimal complexity.

Regularization Impact

- Ridge Regression: Stabilizes the model by penalizing large coefficients, slightly improving performance. Lasso Regression: Performs similarly, but with a marginally higher RMSE (0.05) due to its tendency to shrink some coefficients to zero.

Low Complexity and High Predictive Power

- Low noise and high feature relevance allow these linear models to achieve high accuracy without needing complex non-linear approaches.

Data Forecasting

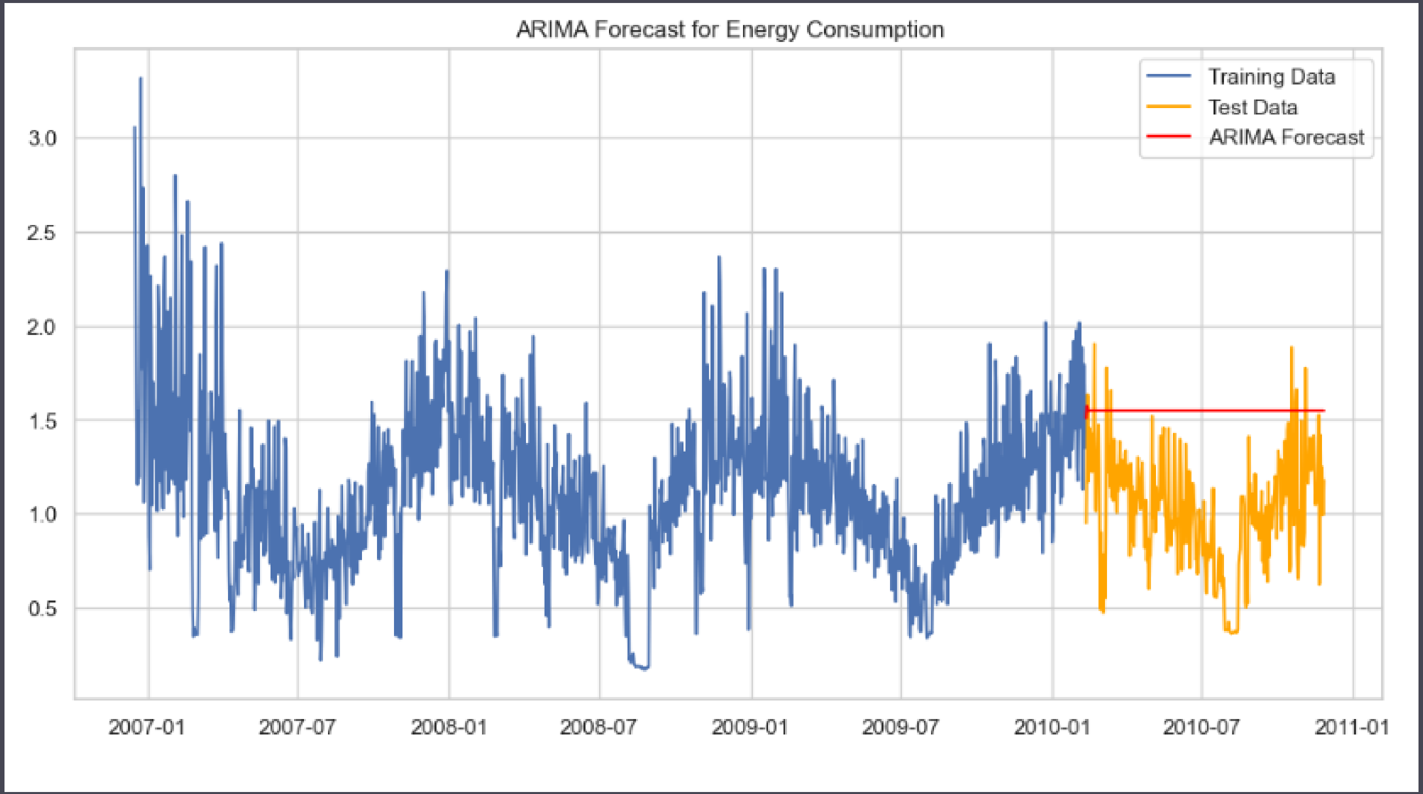
ARIMA :

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends.

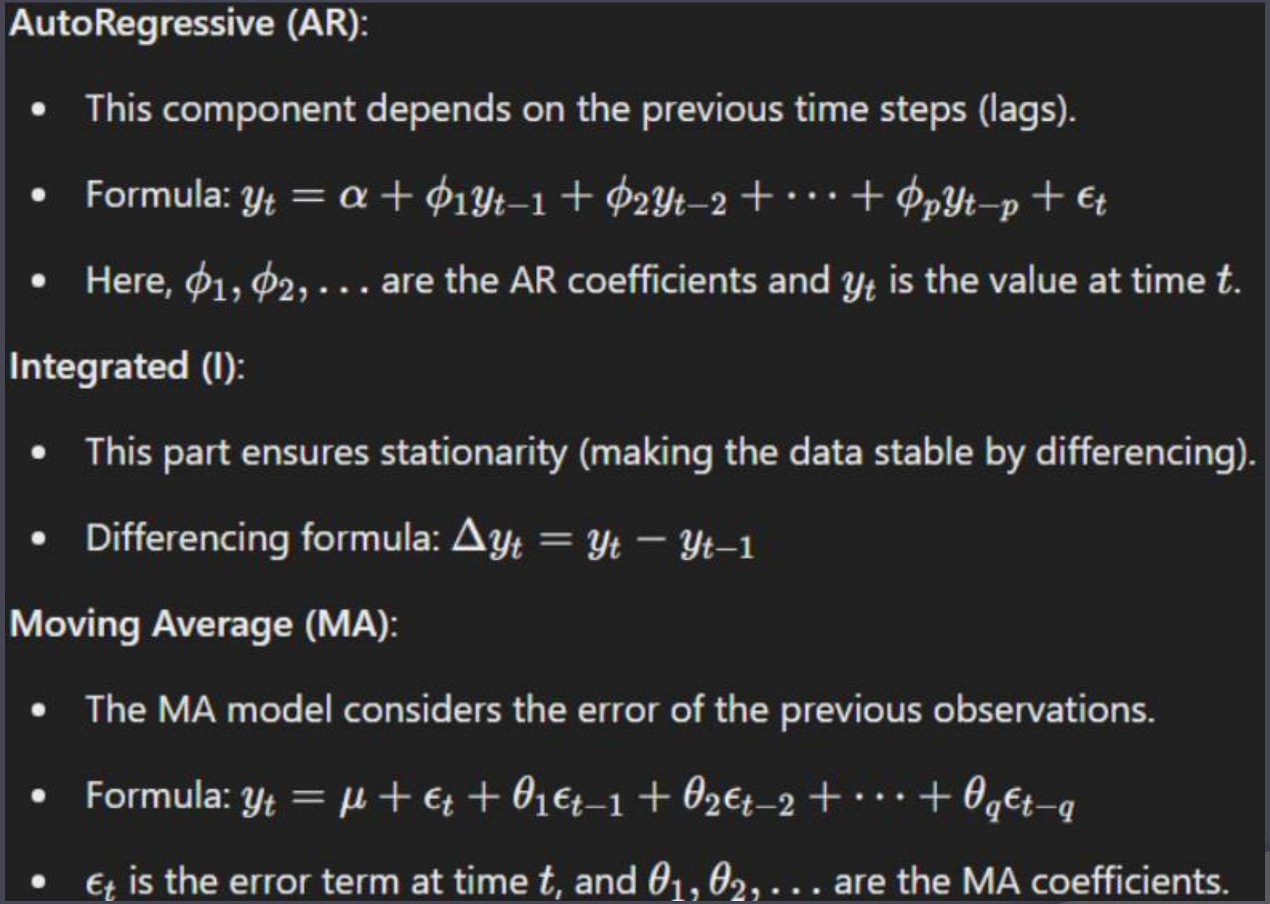
ARIMA Model for Energy Consumption Forecasting

- Aggregate daily energy consumption (Global_active_power).
- Split data: 80% for training, 20% for testing.
- Train ARIMA with parameters (p=2, d=1, q=2)
- Use ARIMA to predict for the test period
- Visualize training data, test data, and predictions to assess performance.

- ARIMA works well for time series data that shows stationarity after differencing, and it captures patterns based on past values and errors.



Math Behind ARIMA



AutoRegressive (AR):

- This component depends on the previous time steps (lags).
- Formula: $y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$
- Here, $\phi_1, \phi_2, \ldots$ are the AR coefficients and y_t is the value at time t.

Integrated (I):

- This part ensures stationarity (making the data stable by differencing).
- Differencing formula: $\Delta y_t = y_t - y_{t-1}$

Moving Average (MA):

- The MA model considers the error of the previous observations.
- Formula: $y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$
- ϵ_t is the error term at time t, and $\theta_1, \theta_2, \ldots$ are the MA coefficients.

Prophet

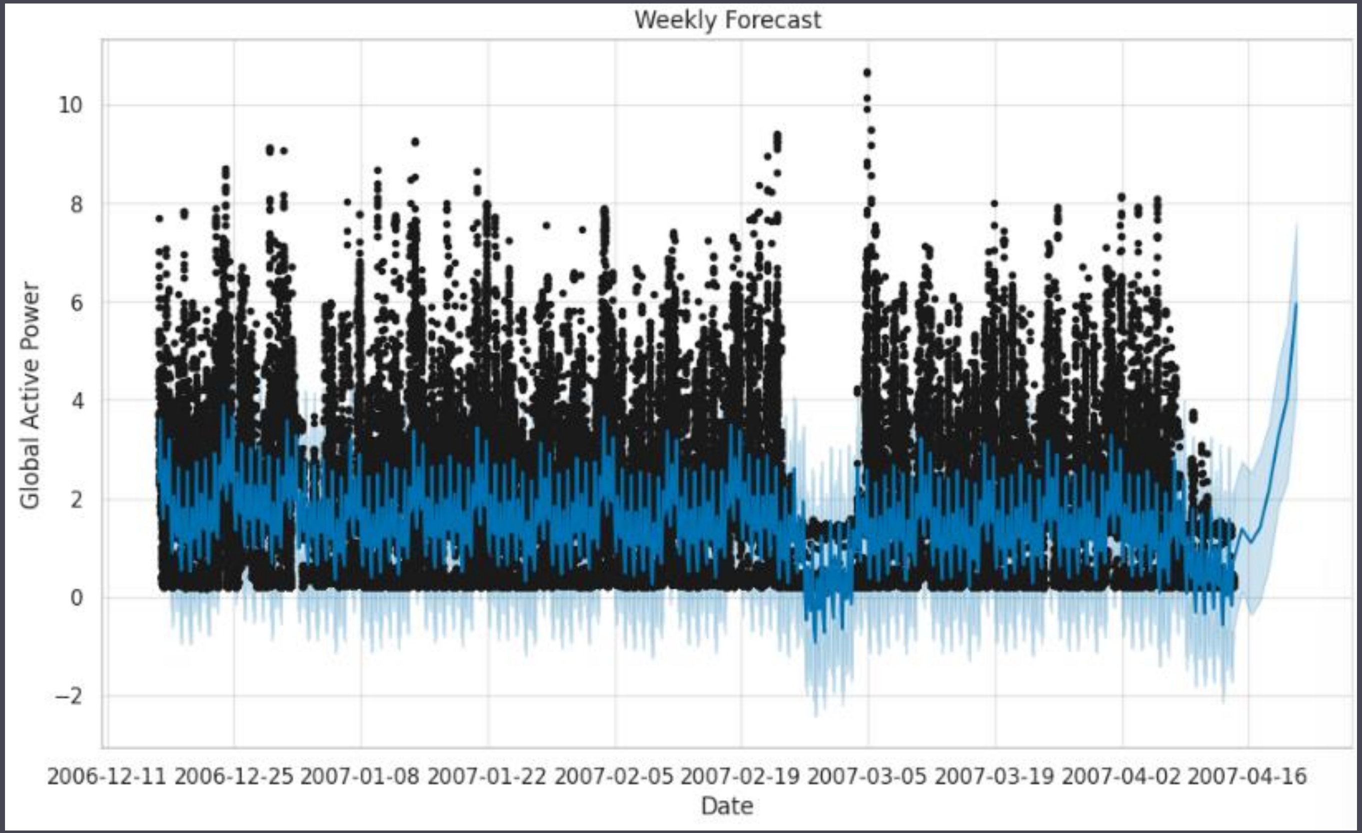
Prophet :

- Prophet is an additive regression model that uses a piecewise linear or logistic growth curve trend to forecast data. It's designed to work best with time series that have strong seasonal effects and several seasons of historical data.

Energy Consumption Forecasting with Prophet

- The dataset is reformatted for Prophet (ds for dates, y for values) and split into training (80%) and testing (20%).
- A Prophet model is configured with yearly, weekly, and daily seasonality, then trained on the training data.
- Forecasts are generated for 7 days (weekly), 30 days (monthly), and 365 days (yearly) into the future.

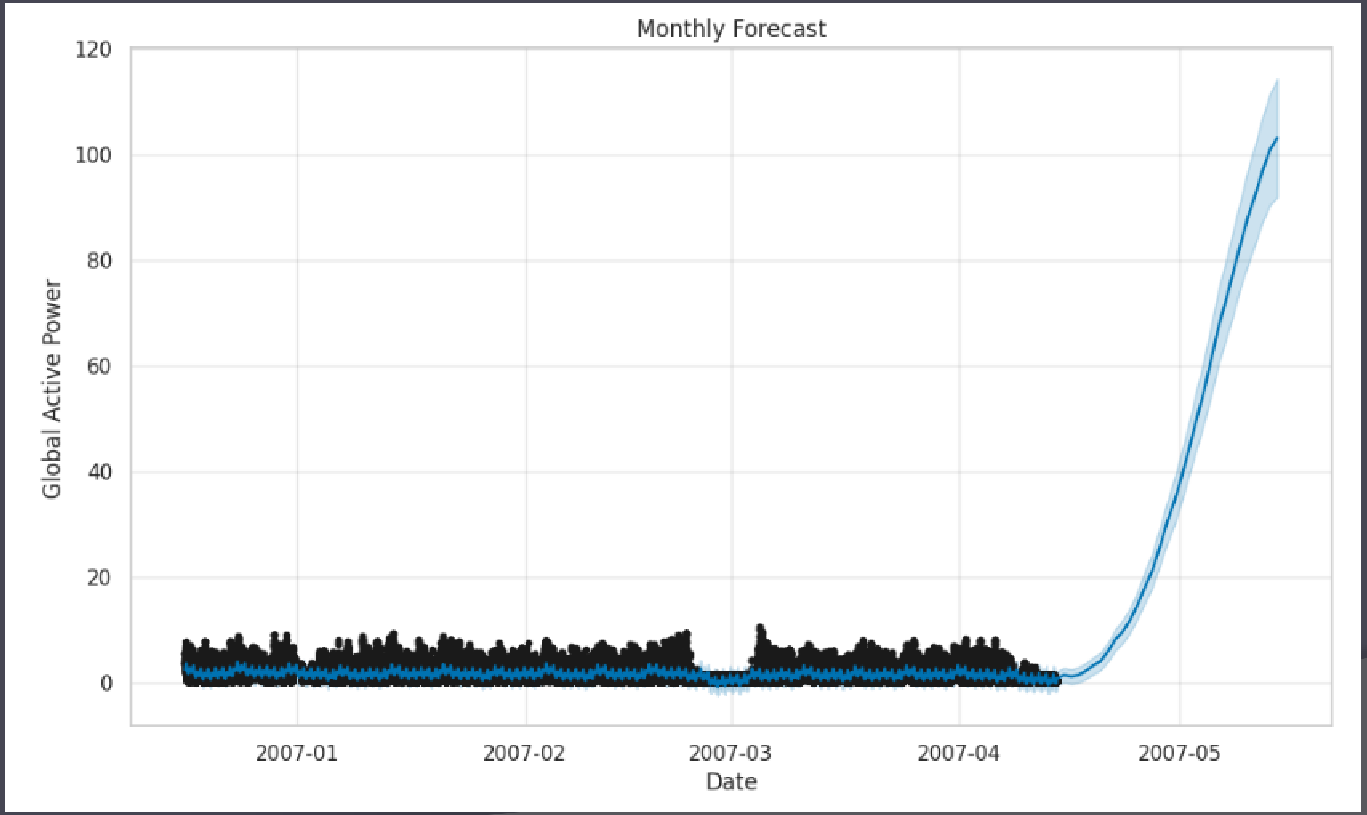
Weekly Forecast



- Weekdays showed higher energy consumption compared to weekends due to typical household or industrial usage patterns.
- Peak usage times within the week were accurately captured.

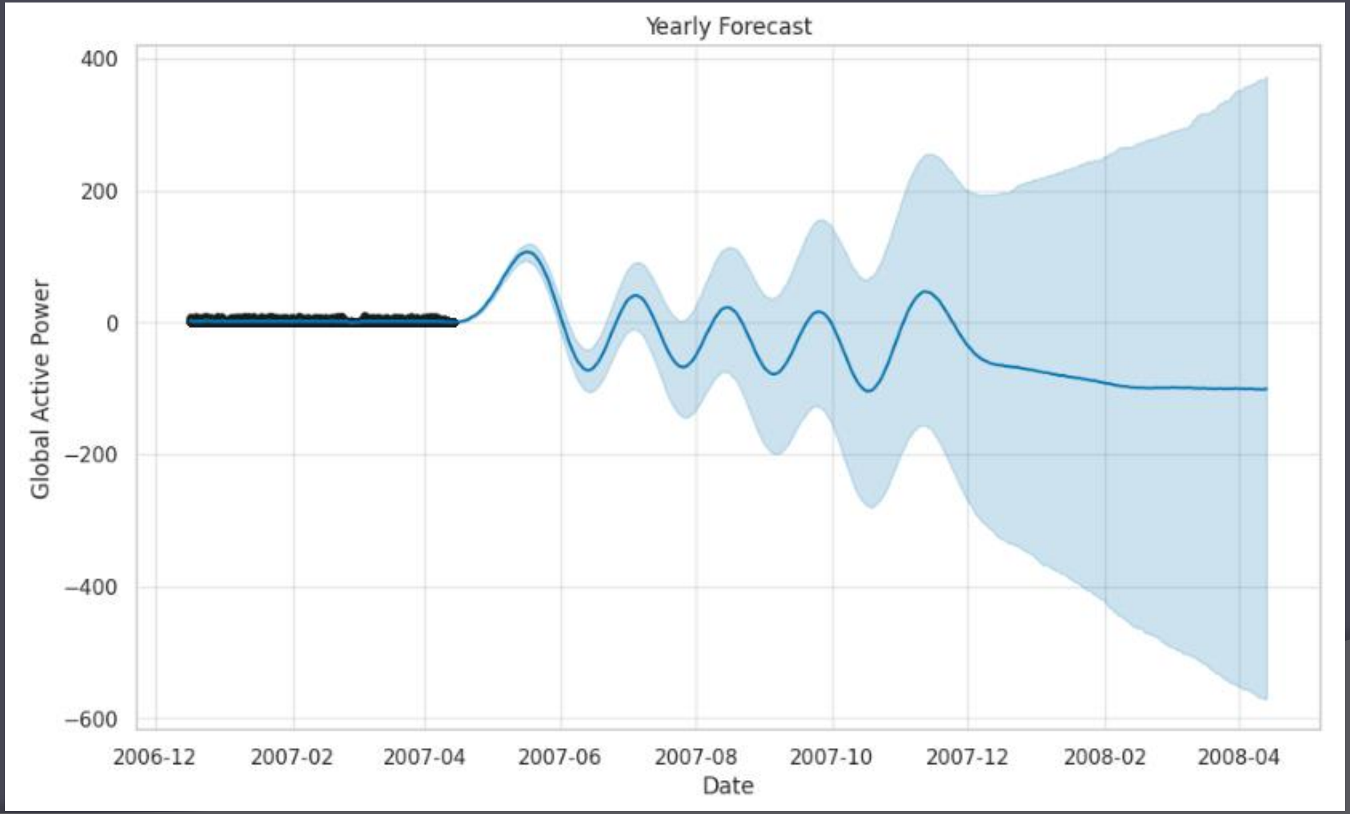
Monthly Forecast

- The forecast captured recurring patterns such as slight dips in consumption on weekends and increased usage mid-month.
- Monthly patterns often reflected cumulative effects of weather and social factors.



Yearly Forecast

- Energy usage increased in colder months due to heating requirements and decreased in warmer months.
- Major holidays, such as festivals or public holidays, showed sharp dips or spikes depending on the nature of energy use.



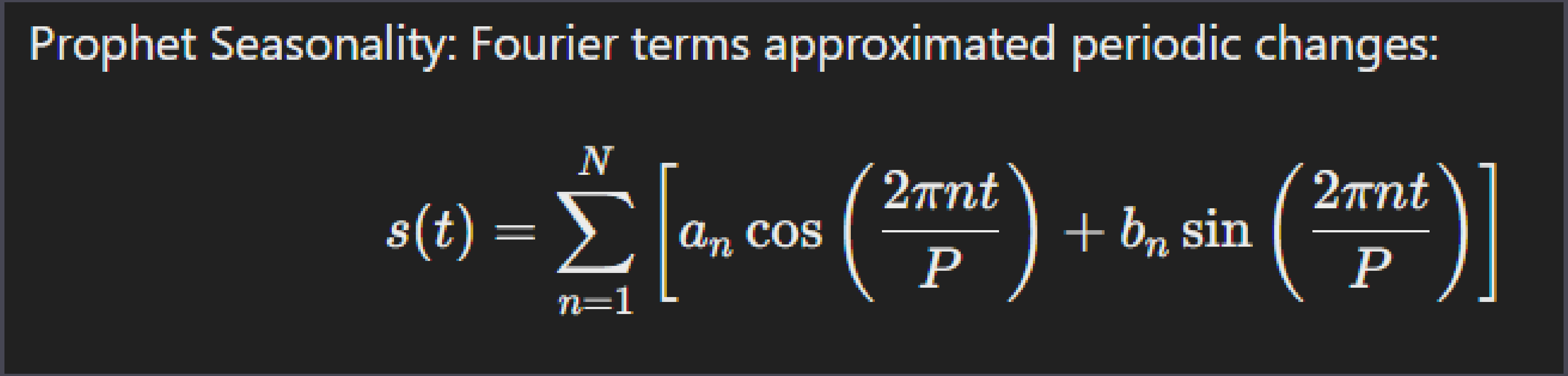
Trend: Indicates long-term growth or decline in energy consumption.

Seasonality:

Yearly: Captures recurring patterns over the years.

Daily: Reflects diurnal energy usage.

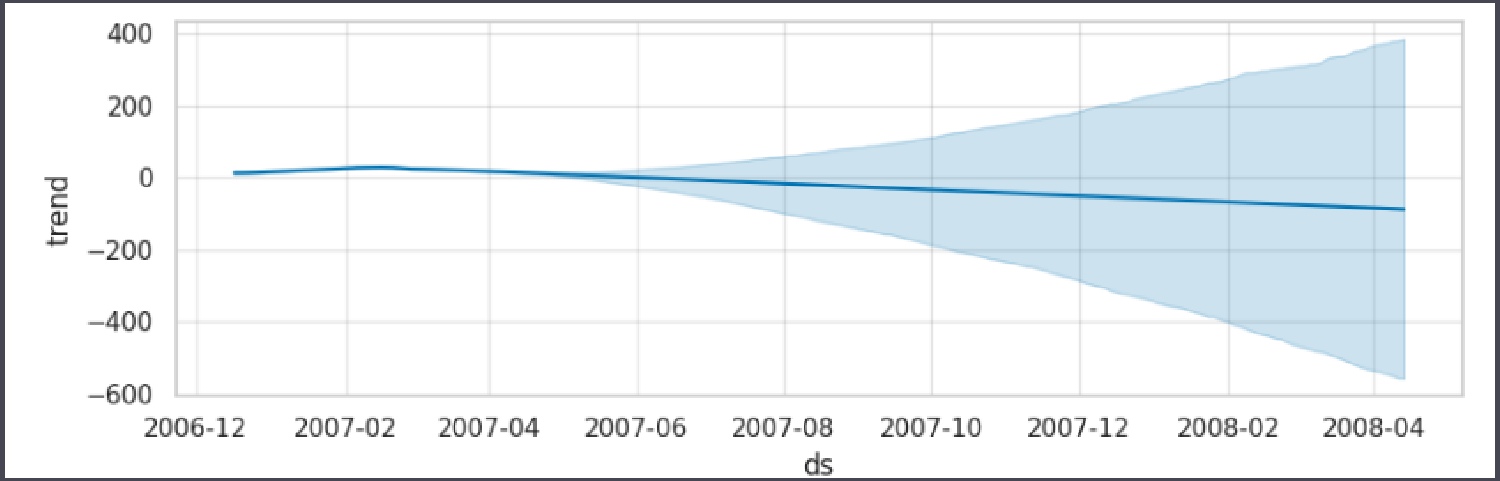
Patterns: Peaks during work hours, dips during off-hours.

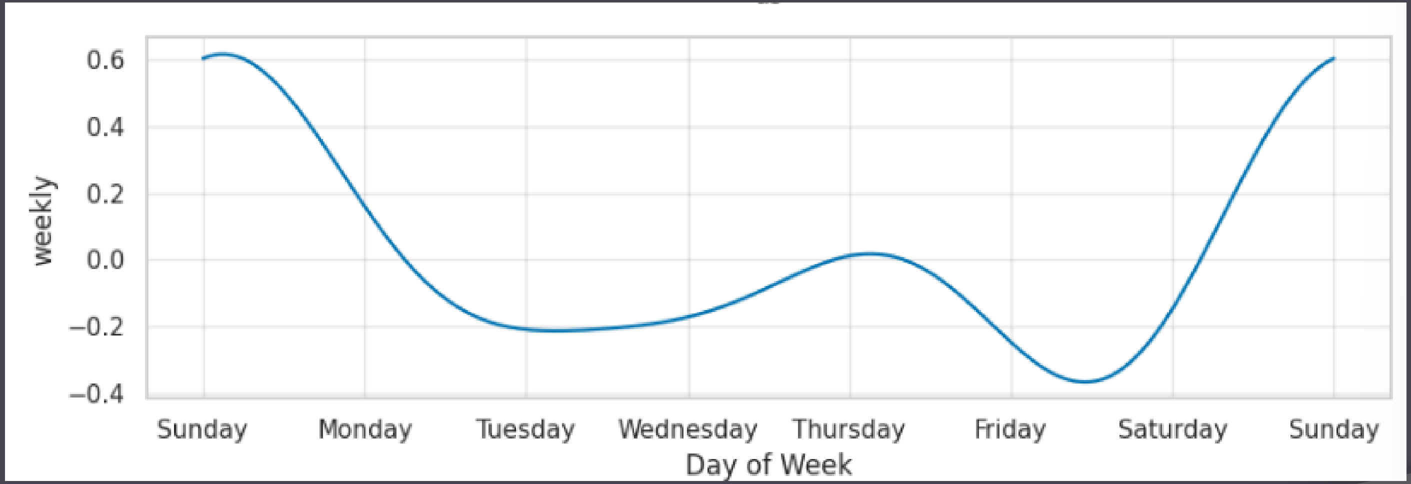


Prophet Seasonality: Fourier terms approximated periodic changes:

$$s(t) = \sum_{n=1}^{N} \left[a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right]$$

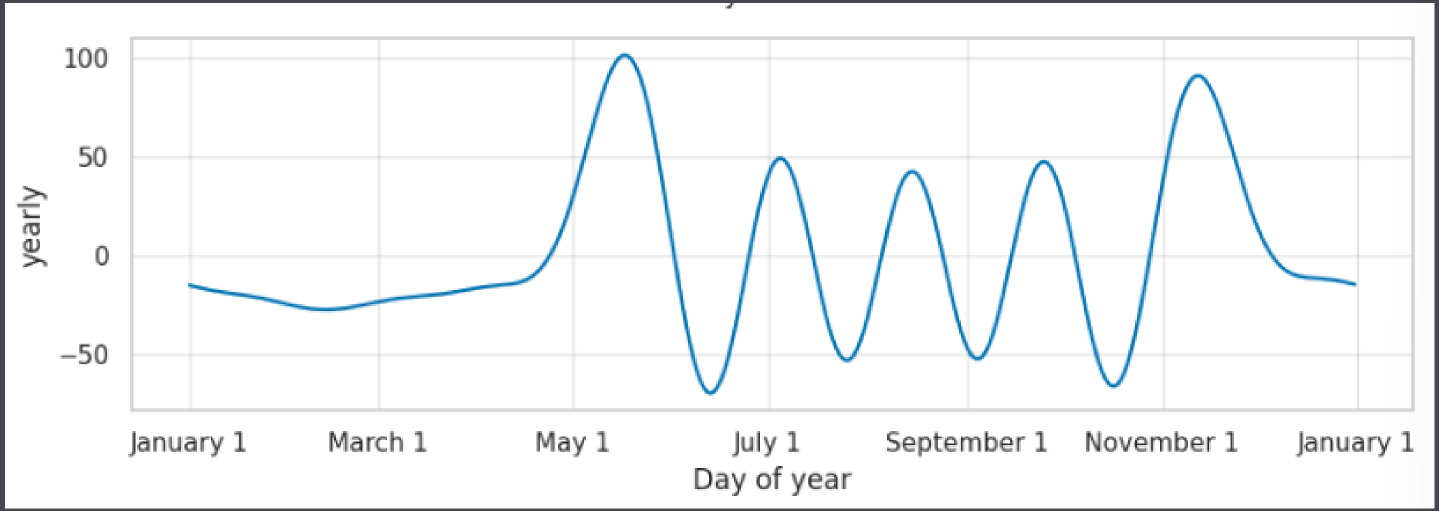
Time-Series Trends:



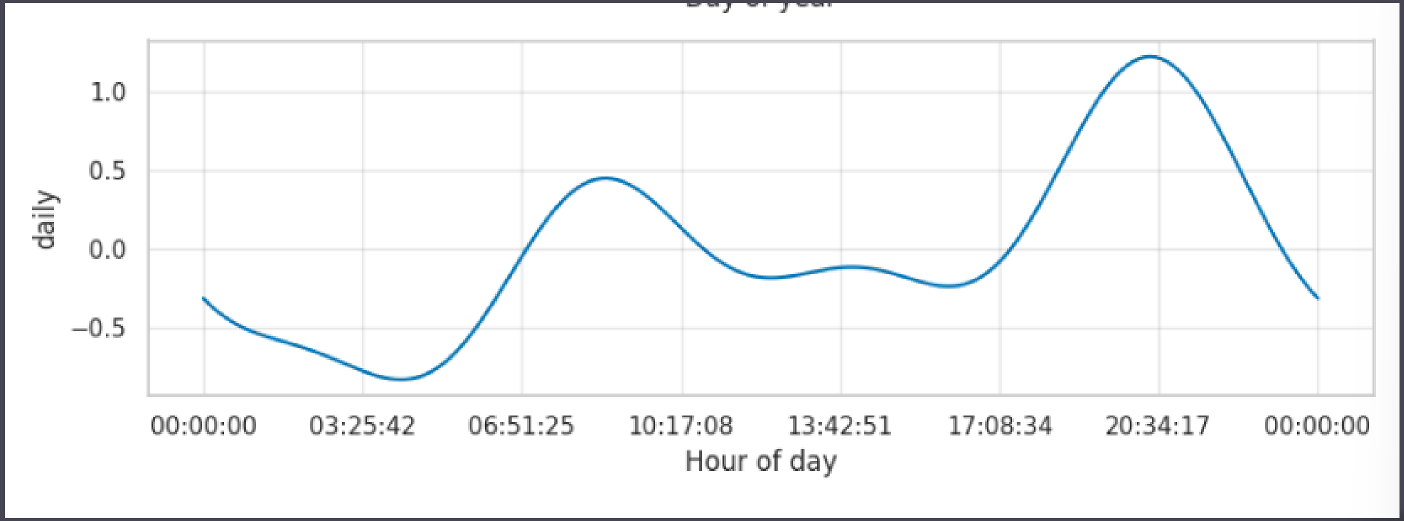


- A gradual upward or downward trend indicates increasing or decreasing energy demands.
- Seasonal dips or spikes may correlate with significant changes in weather, infrastructure, or population habits.

- Higher usage observed on weekdays, often due to industrial or commercial activities
- Lower energy consumption during weekends when residential activity dominates.c



- Peaks in winter months (higher heating needs) or summer months (air conditioning)
- Significant dips during major holidays, such as national festivals or vacation periods.



- Energy consumption gradually rises as people finish work and engage in home activities like cooking, lighting, and entertainment, peaking at **21:00**.

Conclusion

- In this analysis, we have successfully cleaned and visualized energy consumption data to uncover key insights about usage patterns. By exploring hourly, daily, and seasonal trends, we identified significant consumption peaks, particularly in the evening hours, and highlighted the influence of factors like sunlight and holidays. Through robust model building, including Linear, Lasso, and Ridge regression, we evaluated the accuracy of predictions using metrics like RMSE and R^2, with Ridge regression offering a stable performance under multicollinearity.
- forecasting techniques like ARIMA and Prophet, which allowed us to predict future energy consumption trends across weekly, monthly, and yearly intervals, enhancing our understanding of long-term energy demand. These insights can guide better energy management, helping utilities optimize grid operations and improve demand-side management strategies