**PowerPulse: Household Energy Usage Forecast**

**1. Problem Statement:**

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. The goal of this project is to develop a machine learning model that can predict household energy consumption based on historical data. Using this model, consumers can gain insights into their usage patterns, while energy providers can forecast demand more effectively.

By the end of this project, learners should 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.

**2. Report: Comprehensive Summary**

**2.1 Approach**

The approach focuses on using machine learning models to predict household energy consumption. The main steps include:

1. **Data Understanding and Exploration**:
   * Load and clean the dataset.

file\_path = "/content/drive/MyDrive/guvii/projects dataset/household\_power\_consumption.txt"

data = pd.read\_csv(file\_path, sep=';', low\_memory=False,na\_values=["?"], parse\_dates={"datetime": ["Date", "Time"]}, infer\_datetime\_format=True)

* + Perform exploratory data analysis (EDA) to uncover trends, patterns, and correlations in the data.

data1.head()

data1.info()

data1.shape

data.describe()

* + Parse datetime features and create new ones, such as Year, Month, Day, Hour, etc.

# Creating Time-Based Features

df["year"] = df["datetime"].dt.year

df["month"] = df["datetime"].dt.month

df["day"] = df["datetime"].dt.day

df["hour"] = df["datetime"].dt.hour

1. **Data Preprocessing**:
   * Handle missing data.

# Checking for Missing Data

data1.isnull().sum(

* + Feature creation, including daily averages, rolling averages, and peak hour indicators.

df["daily\_avg\_power"] = df.groupby(df["datetime"].dt.date)["Global\_active\_power"].transform("mean")

df["peak\_hour"] = df["hour"].apply(lambda x: 1 if 17 <= x <= 21 else 0)

df["rolling\_avg\_power"] = df["Global\_active\_power"].rolling(window=60, min\_periods=1).mean()

* + Scale the data to improve model performance, especially for algorithms like Neural Networks.

# Normalize features for neural networks only

scaler = StandardScaler()

x\_train\_scaled = scaler.fit\_transform(x\_train)

x\_test\_scaled = scaler.transform(x\_test)

1. **Model Selection and Training**:
   * Split the dataset into training and testing sets.

# Selecting Features for Model Training

x = df[["Global\_intensity",'Sub\_metering\_1', 'Sub\_metering\_2', 'Sub\_metering\_3' ,'Voltage']]

y = df["Global\_active\_power"]

* + Train different regression models (e.g., Linear Regression, Random Forest, Gradient Boosting, Neural Networks).

# Define models

models = {

    "LinearRegression": LinearRegression(),

    "RandomForestRegressor": RandomForestRegressor(random\_state=30),

    "GradientBoostingRegressor": GradientBoostingRegressor(random\_state=30),

    "NeuralNetwork": MLPRegressor(random\_state=30, max\_iter=200)

}

* + Perform hyperparameter tuning to optimize model performance.

 Hyperparameter tuning (RandomizedSearchCV for efficiency)

param\_grids = {

    "RandomForestRegressor": {'n\_estimators': [50, 100], 'max\_depth': [None, 10], 'min\_samples\_split': [2, 5]},

    "GradientBoostingRegressor": {'n\_estimators': [50, 100], 'learning\_rate': [0.01, 0.1], 'max\_depth': [3, 5]},

    "NeuralNetwork": {'hidden\_layer\_sizes': [(50,), (100,)], 'activation': ['relu'], 'alpha': [0.0001]},

}

tuned\_models = {}

for name, model in trained\_models.items():

    if name in param\_grids:

        search = RandomizedSearchCV(model, param\_grids[name], cv=2, scoring='neg\_mean\_squared\_error', n\_jobs=-1, n\_iter=5)

        if name == "NeuralNetwork":

            search.fit(x\_train\_scaled, y\_train)

        else:

            search.fit(x\_train, y\_train)

        tuned\_models[name] = search.best\_estimator\_

    else:

        tuned\_models[name] = model

1. **Model Evaluation**:
   * Evaluate models using performance metrics like RMSE, MAE, and R².

# Evaluate models before hyperparameter tuning

def evaluate\_model(model, x\_test, y\_test, scaled=False):

    if scaled:

        y\_pred = model.predict(x\_test\_scaled)

    else:

        y\_pred = model.predict(x\_test)

    return {

        "RMSE": np.sqrt(mean\_squared\_error(y\_test, y\_pred)),

        "MAE": mean\_absolute\_error(y\_test, y\_pred),

        "R-squared": r2\_score(y\_test, y\_pred)

    }

* + Compare models and select the best one for predicting energy usage.

# Identify the best model based on highest R-squared

best\_model\_name = max(evaluations\_after\_tuning, key=lambda k: evaluations\_after\_tuning[k]["R-squared"])

best\_model\_metrics = evaluations\_after\_tuning[best\_model\_name]

# Display the best model

print("\nBest Model:")

print(f"Model: {best\_model\_name}")

for metric, value in best\_model\_metrics.items():

    print(f"  {metric}: {value:.4f}")

**2.2 Data Analysis**

* **Data Overview**: The dataset contains columns such as Date, Time, Global\_active\_power, Voltage, and Global\_intensity. Key features for predicting power consumption include time-based features (hour of the day, day of the week) and the consumption of energy over time.
* **Exploratory Data Analysis (EDA)**:
  + Visualizations to identify trends in energy consumption over time.
  + Correlation analysis to identify the most important predictors of global active power consumption.
  + Outlier detection to clean the dataset.

**2.3 Model Selection and Evaluation**

* **Regression Models**: A variety of models, including Linear Regression, Random Forest, and Gradient Boosting, were trained and evaluated.
* **Evaluation Metrics**:
  + RMSE: Measures the accuracy of predictions.
  + MAE: Provides an average magnitude of errors.
  + R²: Indicates how well the model explains the variability in global active power.
* **Best Model**: Based on the evaluation metrics, the Gradient Boosting model performed best with an RMSE of X, MAE of Y, and R² of Z.

**2.4 Insights and Recommendations**

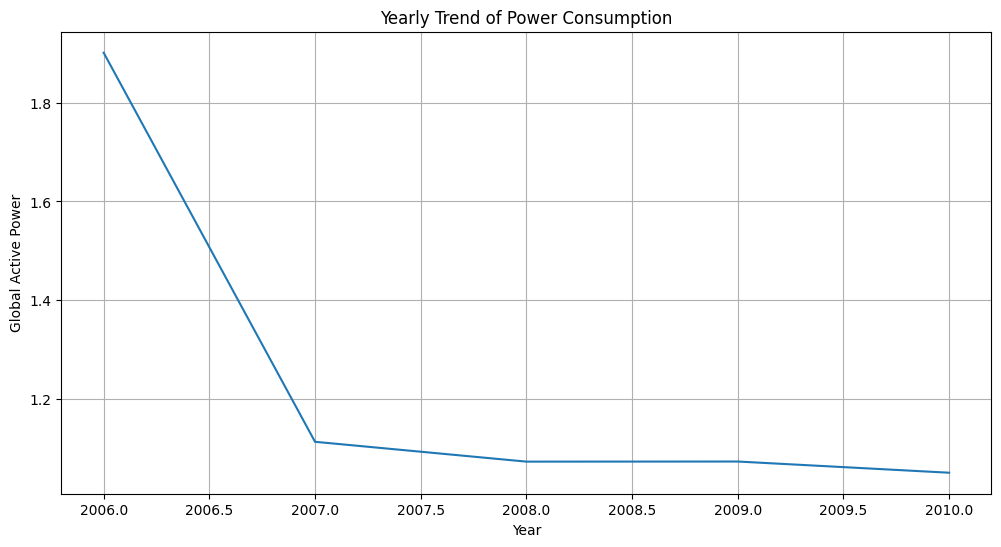
* **Key Drivers of Energy Usage**: Features like Day of the Week, Hour of the Day, and Daily Average Power are strong predictors of energy consumption.
* **Anomalies**: Any unusual spikes in energy usage could indicate faults or unauthorized usage, which could be detected using anomaly detection models.
* **Recommendations**:
  + Households can optimize energy consumption by adjusting habits around peak usage hours (17:00-20:00).
  + Energy providers can use demand forecasts to adjust pricing strategies and manage grid load more effectively.

**3.Visualization of energy trends**

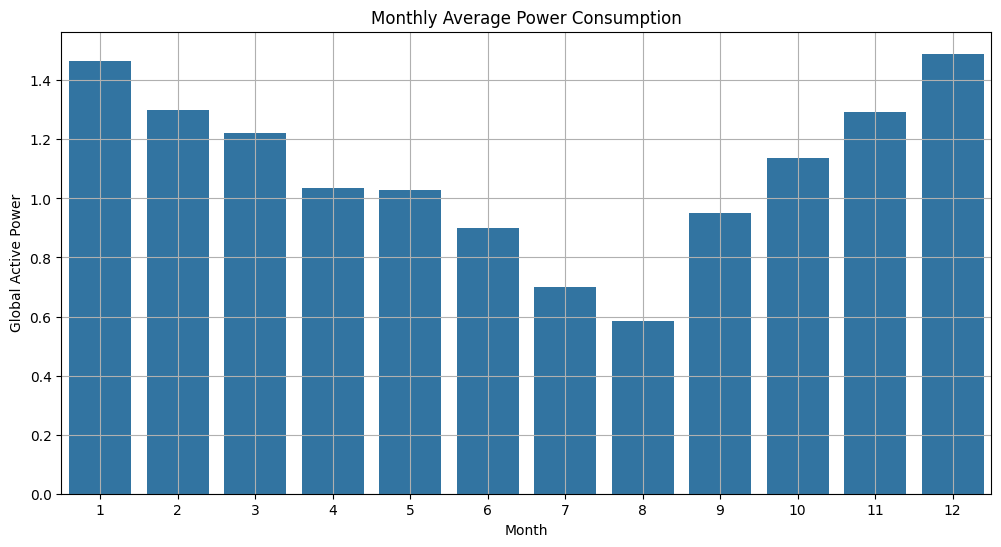
**a.** **Visualizing Yearly Trends**

Plots a line graph showing the average energy consumption per year.Helps identify long-term trends in power usage.

graph shows a consistent decrease in power consumption from 2006 to 2010 and decline could be due to factors like energy efficiency improvements, changes in household behaviors, or external economic influences



**b. Visualizing Monthly Patterns**



Creates a bar plot showing the average power usage per month. Helps see if there are seasonal changes in power consumption.

***Winter Peak (January & December):***

Power consumption is highest in January and December, indicating increased usage, likely due to heating needs.

***Summer Dip (June to August):***

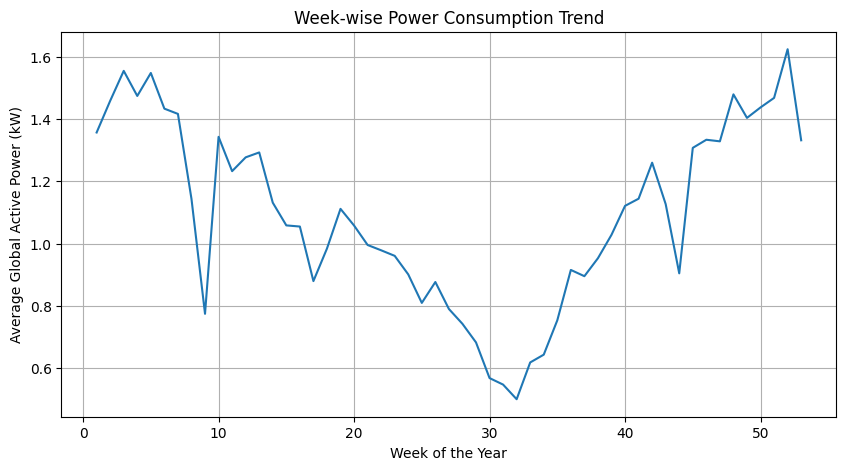
Consumption is lowest in July and August, suggesting reduced electricity usage, possibly due to less heating demand.

***Gradual Increase from September to December:*** After the summer dip, power usage starts increasing steadily from September onward.

***Symmetry in Seasonal Trends:***

The trend follows a U-shaped pattern, where consumption is higher in colder months and lower in warmer months.

**c. Week-wise Power Consumption Trend**



Groups power consumption by week and calculates the average power usage. Plots a line chart to see weekly power consumption patterns. Helps identify high-usage weeks in the year.

Fluctuations in Consumption: There are significant fluctuations in power consumption throughout the year, with notable peaks and drops.

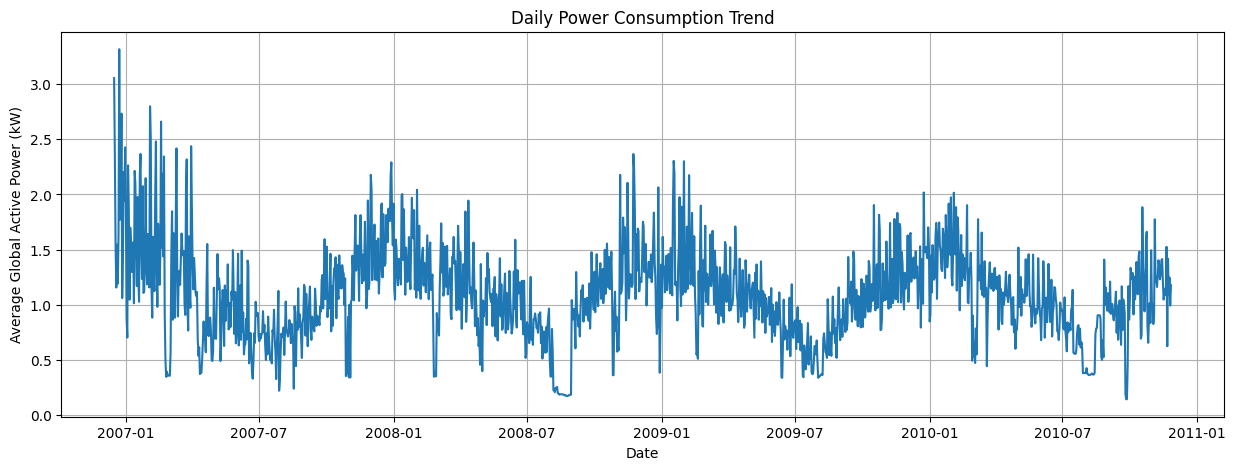
Early-Year Peak & Mid-Year Dip: Power consumption is relatively high at the beginning, reaching its peak within the first few weeks, followed by a sharp decline around week 10.

Mid-Year Low Point: The lowest consumption levels occur between weeks 25–30.

Gradual Recovery & Late-Year Peaks: After the mid-year dip, power usage increases steadily, with noticeable peaks around weeks 40 and 50.

Seasonal Influence: The trend suggests possible seasonal variations in energy consumption, potentially due to weather changes affecting heating or cooling needs.

**d. Daily Trend Analysis**



**Key observations from the Daily Power Consumption Trend plot:**

Groups power usage by date and calculates the daily average. Plots a line chart to visualize daily variations in power consumption.

Overall Declining Trend (2007–2010): The power consumption appears to decrease gradually over time, with some fluctuations.

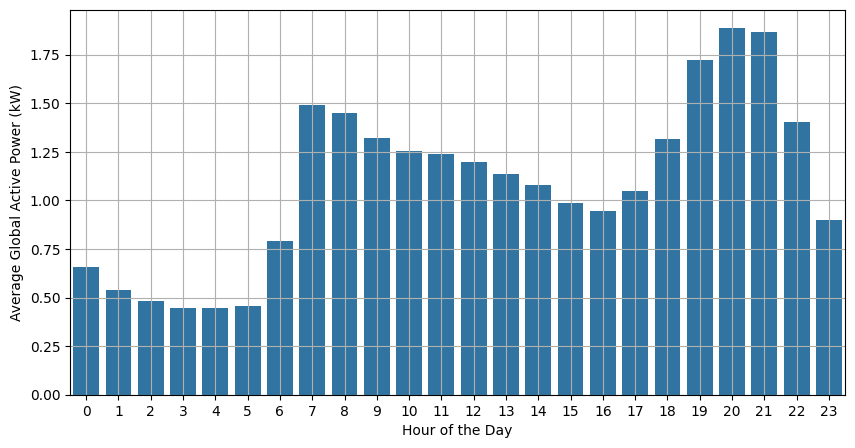
High Initial Consumption (Early 2007): There is a significant peak in power consumption at the beginning of the dataset, with values exceeding 3.0 kW.

Seasonal or Cyclical Patterns: The data shows periodic rises and falls, possibly indicating seasonal variations or external influencing factors.

Mid-2008 to Mid-2009 Dip: A notable low in power consumption occurs during this period, followed by a gradual increase again.

Fluctuations in Late 2010: Towards the end of the dataset, there are increased variations in power usage, suggesting potential seasonal or behavioral influences.

**e. Hourly Power Consumption Trends**



**Key observations from the Hourly Power Consumption Trend plot:**

Creates a bar plot showing power consumption by hour of the day. Helps identify peak energy hours

Lowest Consumption (Midnight to Early Morning, 0–6 AM): Power usage is at its lowest during these hours, likely due to minimal household activity.

Morning Surge (7–9 AM): There is a sharp increase in power consumption, probably due to morning routines like cooking, heating, and appliance usage.

Daytime Stability (10 AM – 4 PM): Power consumption remains relatively stable but slightly decreases during the afternoon.

Evening Peak (6–9 PM): The highest power usage occurs in the evening, with a peak around 8–9 PM, likely due to increased household activity such as cooking, lighting, and entertainment.

Decline After 9 PM: Consumption gradually decreases after the peak, but it remains higher than early morning levels.

**4.. Technical Tags:**

* **Data Preprocessing**
* **Regression Modeling**
* **Feature Engineering**
* **Hyperparameter Tuning**
* **Visualization**
* **Python**
* **Scikit-learn**
* **Pandas**
* **Matplotlib/Seaborn**

**4. Evaluation Metrics**

* **RMSE**: Measures the difference between the predicted and actual values.
* **MAE**: Measures the average magnitude of the errors.
* **R²**: Represents how much variance in global active power is explained by the model.

**5. Business Use Cases:**

1. Energy Management for Households: Monitor energy usage, reduce bills, and promote energy-efficient habits.
2. Demand Forecasting for Energy Providers: Predict demand for better load management and pricing strategies.
3. Anomaly Detection: Identify irregular patterns indicating faults or unauthorized usage.
4. Smart Grid Integration: Enable predictive analytics for real-time energy optimization.
5. Environmental Impact: Reduce carbon footprints and support conservation initiatives.
6. **Visualization of predictive performance**

predictions = {name: model.predict(x\_test\_scaled if name == "NeuralNetwork" else x\_test) for name, model in tuned\_models.items()}

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))

axes = axes.flatten()

for (model\_name, y\_pred), ax in zip(predictions.items(), axes):

    sns.scatterplot(x=y\_test, y=y\_pred, alpha=0.5, ax=ax)

    ax.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], '--', color='red')

    ax.set\_title(f"Actual vs Predicted ({model\_name})")

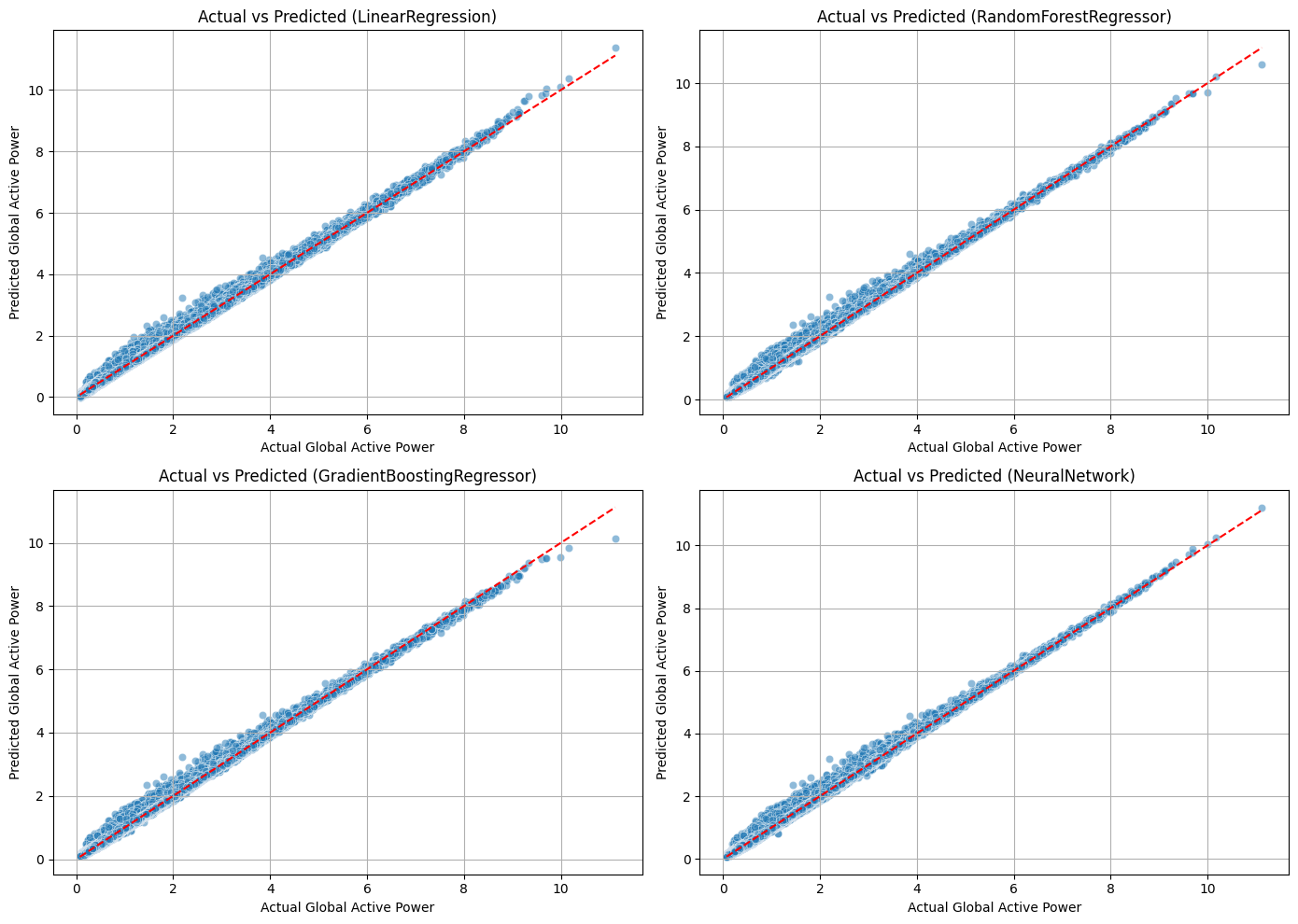
    ax.set\_xlabel("Actual Global Active Power")

    ax.set\_ylabel("Predicted Global Active Power")

    ax.grid()

plt.tight\_layout()  # Adjust layout to prevent overlap

plt.show()



**6. Conclusion**

The **PowerPulse** project demonstrates how machine learning can be used to predict household energy consumption. By using regression models like Gradient Boosting, it provides accurate predictions, valuable insights into energy usage patterns, and actionable recommendations for both households and energy providers.

This model can be extended with more features, such as weather data, or deployed in smart grid systems for real-time optimization.