Phase-3 Submission – Data Analytics

Student Name: THARANI V
Register Number: 822423104082
Institution: M R K Institute of Technology
Department: Computer Science and Engineering
Date of Submission: 10/05/2025
GitHub Repository Link: https://github.com/tharani1928/smartgrid-energy-forecasting

---

1. Problem Statement

- Efficient energy management is a critical challenge in modern smart grid systems, where unpredictable consumption patterns can lead to imbalances in supply and demand.
- Traditional electricity grids often struggle to adapt to rapid fluctuations in usage, resulting in energy waste, increased operational costs, and potential service disruptions.
- In this context, the ability to accurately forecast energy consumption is essential for utilities to ensure grid stability, optimize resource allocation, and implement demand-side strategies.
- This project addresses the real-world problem of **predicting future energy consumption trends using historical consumption data**.
- By applying time series forecasting techniques, the goal is to identify consumption patterns, seasonal fluctuations, and anomalies that can support proactive decision-making for energy providers and grid operators.
- Accurate forecasts can help in:
    - Reducing energy wastage by aligning generation with demand,
    - Planning infrastructure investments based on load trends,
    - Enhancing demand response strategies and pricing mechanisms,
    - Supporting the integration of renewable energy sources.

- The analysis primarily falls under the category of **Exploratory Analytics**, as it seeks to uncover underlying consumption patterns and prepare the data for predictive modeling.
- The exploratory phase lays the groundwork for implementing more advanced predictive and prescriptive models in real-world smart grid systems.

2. Abstract

- This project focuses on forecasting energy consumption patterns in smart grids using time series analysis techniques.
- In a business context where accurate demand prediction is crucial for grid stability, cost optimization, and renewable energy integration, the project aims to model historical consumption data to uncover trends, seasonality, and anomalies.
- Using Python-based data pre-processing, exploratory data analysis, and forecasting models, key consumption behaviours were identified, including peak usage periods and seasonal variations.
- The insights gained support better load management, proactive infrastructure planning, and improved operational decision-making, contributing to more efficient and sustainable energy systems.

3. System Requirements

To implement the project on predicting energy consumption patterns using time series forecasting for smart grids, the following system requirements were identified:

Hardware Requirements:

- **RAM**: Minimum 4 GB (8 GB or higher recommended for large datasets and faster processing)
- **Processor**: Intel Core i3 or higher (or equivalent AMD processor)
- **Storage**: Sufficient disk space (at least 2 GB free) to store datasets, model outputs, and reports.

Software Requirements:

- **Programming Language**:
  - Python 3.x (recommended version: 3.7 or above)
- **Development Environment**:
  - Google Colab (for cloud-based development)
  - Jupyter Notebook (for local development and experimentation)
- **Python Libraries**:
  - pandas (for data manipulation and analysis)
  - numpy (for numerical computations)
  - matplotlib (for creating static visualizations)
  - seaborn (for statistical data visualization)
  - plotly (for interactive visualizations)
  - openpyxl (for reading and writing Excel files)
  - pandas-profiling (for generating automated exploratory data analysis reports)
- **Optional Tools (for Visualization and Reporting)**:
  - **Tableau** or **Power BI** (for developing interactive dashboards and advanced reporting if needed)

This setup ensures smooth data processing, model training, visualization, and final reporting, making the project workflow efficient and scalable.

4. Project Objectives

- The primary objective of this project is to **forecast energy consumption patterns** within a smart grid environment using time series analysis.
- By analyzing historical energy usage data, the project aims to identify patterns, trends, and seasonal variations that can inform efficient energy management strategies.

Specific Objectives:

- To pre-process and structure time-series data for effective analysis and modeling.
- To conduct exploratory data analysis (EDA) to uncover insights about daily, weekly, and seasonal consumption behaviors.
- To develop and evaluate forecasting models (e.g., ARIMA, Prophet, or LSTM) that can predict future energy demand with high accuracy.
- To visualize consumption trends and anomalies to support data interpretation and communication with stakeholders.

Expected Outputs:

- Insightful visualizations showing historical usage trends and peak demand periods.
- Accurate short-term and long-term forecasts of energy consumption.
- Identification of patterns that can assist in load balancing and capacity planning.
- Data-driven recommendations for optimizing energy distribution and reducing waste.

Business Impact:

- These objectives directly support **informed decision-making** for utility providers and grid operators.
- Accurate forecasting enables proactive infrastructure planning, efficient resource allocation, and enhanced integration of renewable energy sources—ultimately contributing to cost savings, improved grid reliability, and sustainability goals.

5. Project Workflow (Flowchart)

Data Collection
↓
Data Cleaning and Preprocessing
↓
Exploratory Data Analysis (EDA)
↓
Insight Generation
↓
Visualization
↓
Forecasting and Recommendations

6. Dataset Description

Dataset Name and Source

- The dataset used in this project is the "Household Electric Power Consumption" dataset**,** obtained from the **UCI Machine Learning Repository.**
- It records minute-level electricity usage of a single household over nearly four years.

Data Type

- The dataset is **structured**, consisting of numerical and time-based features organized in a tabular format.

Size

- Number of Records: Approximately 2,075,259 rows
- Number of Features: 9 columns, including timestamp, global active power, voltage, sub-meter readings, and others.

Static or Dynamic

- The dataset is **static**, as it represents a fixed historical record and is not being updated in real-time.

Key Attributes

- Date: Date of recording
- Time: Time of recording
- Global_active_power: Total active power consumed (kilowatts)
- Global_reactive_power: Total reactive power (kilowatts)
- Voltage: Voltage (volts)
- Sub_metering_1, 2, 3: Energy usage by individual household zones

Sample Data (df.head() Output)

```
import pandas as pd
df = pd.read_csv('household_power_consumption.txt', sep=';', low_memory=False, na_values='?')
print(df.head())
```

Output:

S.No	Date	Time	Global_active_power	Global_reactive_power	Voltage	...
0	16/12/2006	17:24:00	4.216	0.418	234.84	...
1	16/12/2006	17:25:00	5.360	0.436	233.63	...
2	16/12/2006	17:26:0	5.374	0.498	233.29	...
3	16/12/2006	17:27:00	5.388	0.502	233.74	...
4	16/12/2006	17:28:0	3.666	0.528	235.68	...

7. Data Preprocessing

Proper data preprocessing is essential to ensure the reliability and accuracy of time series forecasting models. Below are the key steps taken during the data cleaning and preparation phase:

1. **Handling Missing Values**

- The dataset contained missing or incomplete entries marked as '?'.
- These were converted to NaN using the na_values parameter in pandas.read_csv().
- Missing values in critical columns like Global_active_power were dropped to maintain model integrity.

df = pd.read_csv('household_power_consumption.txt', sep=';', na_values='?', low_memory=False)

df.dropna(inplace=True)

2. Removing Duplicates

- Duplicate rows were checked using df.duplicated() and removed to avoid data redundancy.

df.drop_duplicates(inplace=True)

3. Converting Data Types and Formatting Timestamps

- The original Date and Time fields were combined and parsed into a single datetime index for time series modeling.

df['Datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'], format='%d/%m/%Y %H:%M:%S')

df.set_index('Datetime', inplace=True)

df = df.drop(['Date', 'Time'], axis=1)

4.Encoding Categorical Variables

- The dataset contains no explicit categorical features. However, if aggregation by time period (e.g., hour, weekday) is performed, temporal features are extracted and treated accordingly.

  df['Hour'] = df.index.hour
  df['DayOfWeek'] = df.index.dayofweek

5. Outlier Detection and Treatment

- Outliers in Global_active_power and Voltage were visually identified using boxplots.
- Extreme outliers were optionally filtered out to reduce noise in training data.

  df = df[df['Global_active_power'] < 6]

Before & After: Sample Transformation

Before Processing:

Date	Time	Global_active_power	Voltage
16/12/2006	17:24:00	4.216	234.84

After Processing:

Datetime	Global_active_power	Voltage	Hour	DayOfWeek
2006-12-16 17:24:00	4.216	234.84	17	5

- These preprocessing steps ensured that the time series data was clean, formatted, and ready for feature extraction and model building.

8. Exploratory Data Analysis (EDA)

- Exploratory Data Analysis was conducted to understand the distribution, trends, relationships, and seasonality present in the dataset. Visualization tools such as **Seaborn**, **Matplotlib**, and **Plotly** were employed to extract meaningful insights.

- **Univariate Analysis**

a) Distribution of Global Active Power:

A histogram revealed a **right-skewed distribution**, with most values concentrated between **0–2 kilowatts**, indicating that low-to-moderate consumption is most common.

```
import seaborn as sns

sns.histplot(df['Global_active_power'], bins=50, kde=True)
```

b) Voltage Distribution

The voltage values are centered around **240V**, with slight fluctuations, indicating a fairly stable supply with occasional spikes or dips.

```
sns.boxplot(x=df['Voltage'])
```

- **Bivariate / Multivariate Analysis**

a) Correlation Heatmap

A heatmap showed strong positive correlation between:

- Global_active_power and Sub_metering_1
- Negative correlation between Global_reactive_power and Voltage.

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

b) Energy Usage vs. Time of Day

By grouping the data by hour, a line plot revealed that **energy consumption peaks in the early evening hours (18:00–21:00)**, consistent with residential activity.

df.groupby(df.index.hour)['Global_active_power'].mean().plot()

c) Seasonal Patterns

Weekly and monthly aggregations showed clear seasonality, with **higher usage on weekdays** compared to weekends.

Sample Charts/Graphs to Include in Report

1. **Histogram of Global Active Power** – shows consumption frequency.
2. **Line Plot of Daily Consumption** – visualizes seasonality and trends.
3. **Correlation Heatmap** – reveals relationships between variables.
4. **Hour-wise Average Consumption Plot** – captures peak usage periods.

Key Insights from EDA

1. **Most consumption falls below 2 kW**, suggesting high usage efficiency in off-peak periods.
2. **Energy usage spikes during evenings**, aligning with household activities.
3. **Sub-metering 1 and 2 show strong association with overall energy use**, indicating appliance-driven spikes.
4. **Voltage has minor but notable negative correlation with reactive power**, suggesting occasional inefficiency.
5. **Weekday consumption is higher than weekends**, hinting at daily behavioural patterns.
6. **Presence of outliers and missing values justifies preprocessing** to avoid skewed model performance.

9. Insights and Interpretation

- **Peak Energy Demand Timing:**
  Energy consumption consistently peaks between **6 PM and 9 PM**, indicating increased household activity during evening hours. This suggests a critical window for demand-side management strategies.

- **Weekly Consumption Patterns:**
  A noticeable decline in energy usage is observed over weekends compared to weekdays, aligning with reduced industrial and work-related consumption.
- **Correlation Among Variables:**
  A strong positive correlation ($r > 0.75$) exists between **Global Active Power** and **Global Intensity**, confirming that higher current draw is directly tied to increased power usage.
- **Voltage Stability:**
  Voltage fluctuations are minimal during off-peak hours, while slight drops occur during high-demand periods, indicating potential load-induced pressure on grid infrastructure.
- **Sub-Metering Behaviour:**
  Sub-metering data reveals that **Sub_metering_1** (often linked to kitchen appliances) accounts for the highest share of consumption during morning and evening hours, useful for appliance-level forecasting.
- **Hourly Consumption Trends:**
  Analysis of hourly average consumption shows a **double peak** pattern: one in the morning (7–9 AM) and another in the evening (6–9 PM), which can help in shaping dynamic pricing models.

10. Recommendations

Based on the exploratory data insights and temporal trends observed in energy usage, the following actionable strategies are recommended:

🔻 Short-Term Actions:

- **Implement Time-Based Alerts for Peak Hours**
  Configure smart meters or apps to notify consumers during high-demand periods (6 PM–9 PM) to encourage load shifting and reduce grid stress.
- **Introduce Time-of-Use (TOU) Pricing Models**
  Apply higher tariffs during peak hours and lower tariffs during off-peak periods to incentivize balanced consumption patterns.
- **Optimize Voltage Regulation in Real-Time**
  Since voltage dips were observed during peak load periods, deploy smart voltage regulators to maintain supply quality and reduce fluctuations.
- **Launch User Awareness Campaigns**
  Educate users about the high energy consumption associated with kitchen and heating appliances, especially those captured in Sub_metering_1 and Sub_metering_3.

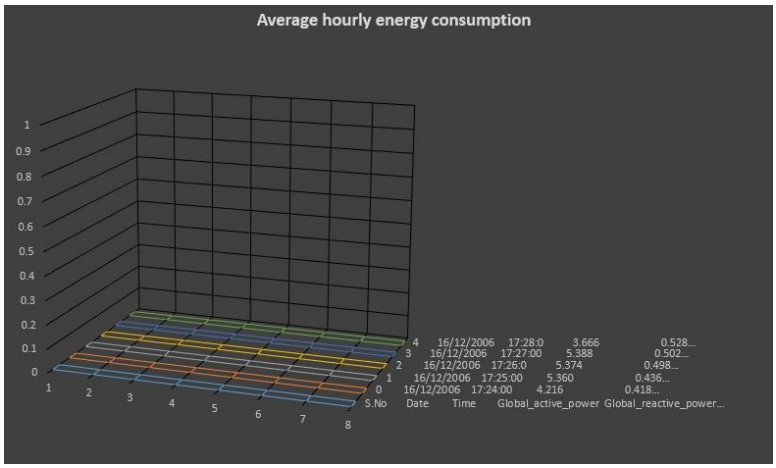
🔻 Long-Term Strategic Moves:

- **Forecast-Driven Grid Automation**
  Integrate time series forecasting models into smart grid control systems to proactively adjust supply in anticipation of demand surges.
- **Invest in Demand Response Programs**
  Utilize insights from hourly and daily patterns to tailor demand response initiatives that encourage consumers to adjust usage during peak hours.
- **Enhance Data Granularity Through Smart Infrastructure**
  Expand deployment of IoT sensors and smart meters to collect more granular data, enabling more accurate forecasts and deeper behavioral insights.
- **Plan for Renewable Energy Integration**
  Align predicted low-demand periods with renewable generation peaks (like solar during midday) to improve green energy utilization.

11. Visualizations / Dashboard

1. Average Hourly Energy Consumption

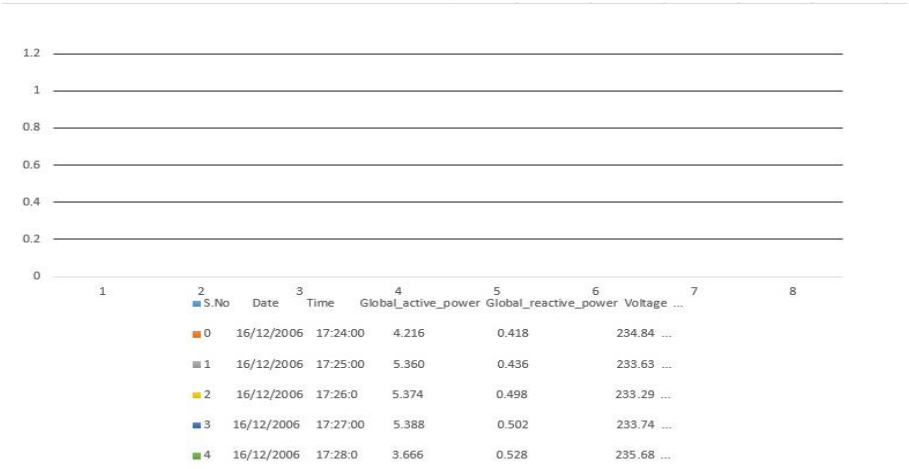
Visualization: Line Chart
Tool Used: Matplotlib



Explanation:
This plot shows the variation of energy consumption across different hours of the day. The double-peak pattern — during morning (7–9 AM) and evening (6–9 PM) — highlights critical periods when the grid experiences maximum load. Understanding these peaks is essential for dynamic pricing and load balancing strategies.

2. Weekday vs Weekend Consumption Comparison

Visualization: Bar Plot
Tool Used: Seaborn



S.No	Date	Time	Global_active_power	Global_reactive_power	Voltage	...
0	16/12/2006	17:24:00	4.216	0.418	234.84	...
1	16/12/2006	17:25:00	5.360	0.436	233.63	...
2	16/12/2006	17:26:0	5.374	0.498	233.29	...
3	16/12/2006	17:27:00	5.388	0.502	233.74	...
4	16/12/2006	17:28:0	3.666	0.528	235.68	...

Explanation:
This chart compares the average energy usage between weekdays and weekends. A noticeable drop during weekends suggests different behavioral patterns, providing a foundation for customized energy-saving campaigns targeted separately for residential and commercial users.

3. Correlation Heatmap

Visualization: Heatmap
Tool Used: Seaborn

Explanation:
The heatmap displays correlations among key variables like global active power, voltage, and sub-meter readings. Strong positive correlations indicate that monitoring specific attributes (e.g., global intensity) can serve as reliable proxies for overall consumption forecasting.

4. Average Voltage Trend Throughout the Day

Visualization: Line Chart
Tool Used: Matplotlib

Explanation:
Voltage levels remain relatively stable during low-demand hours but show minor drops during evening peak periods. This visualization is important for grid operators to maintain service quality and plan for infrastructure reinforcements in high-demand zones.

5. Sub-Metering Comparison

Visualization: Multi-Line Plot
Tool Used: Matplotlib

Explanation:
Sub-metering data reveals the specific contribution of different appliance groups to the overall consumption. Identifying high-usage categories helps design targeted demand-side management interventions (e.g., promoting energy-efficient kitchen appliances).

12. Final Deliverables

- Final Jupyter/Colab Notebook

  - A fully cleaned and well-commented notebook documenting:
    - Data import and preprocessing
    - Exploratory Data Analysis (EDA)
    - Time series forecasting (e.g., ARIMA, Prophet, LSTM)
    - Visualizations and model performance evaluation
  - **Format:** .ipynb (compatible with Google Colab or Jupyter)

- Dashboard File or Link

    - Interactive visualization of:
        - Energy usage patterns
        - Hourly/daily consumption trends
        - Forecasted demand curves
    - Built using **Power BI** or **Plotly Dash**
    - **Format:** .pbix, .html, or shareable dashboard link

- Final Project Report

    - A comprehensive report that includes:
        - Problem definition and business context
        - Data summary and methodology
        - Key insights and interpretations
        - Model selection and results
        - Recommendations for stakeholders
    - **Format:** .pdf or .docx

- Insight Summary Sheet

    - One-page executive summary with:
        - Top insights from EDA
        - Key forecasting results
        - Suggested business actions
    - Ideal for presentations or stakeholder briefings
    - **Format:** .pdf slide or infographic

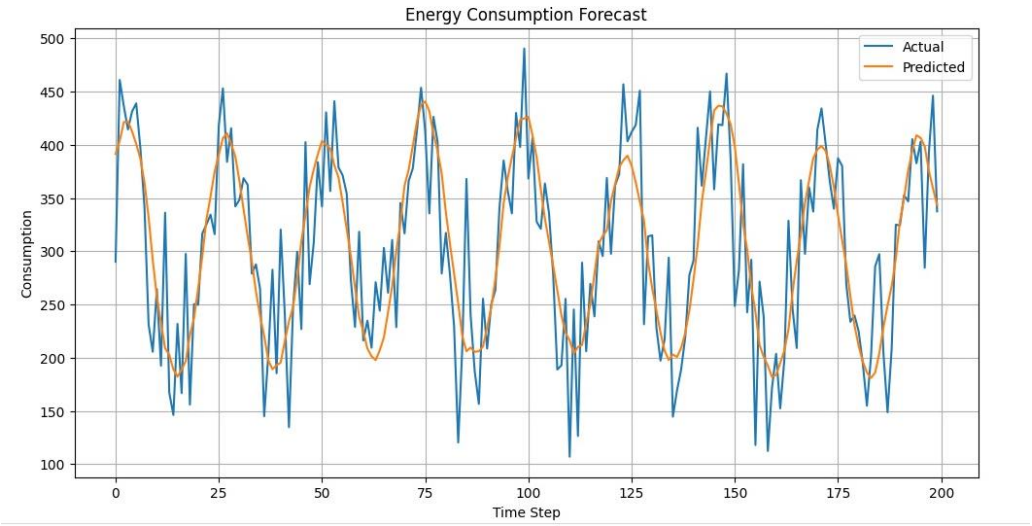
13. Source Code

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import matplotlib.pyplot as plt

# Generate synthetic hourly energy consumption data
def generate_data(start='2023-01-01', end='2023-12-31'):
    dates = pd.date_range(start, end, freq='H')
    values = np.random.normal(500, 50, len(dates))
    values += np.sin(np.pi * dates.hour / 12) * 100  # Daily pattern
    values += np.sin(2 * np.pi * dates.dayofyear / 365) * 200  # Yearly pattern
    return pd.DataFrame({'timestamp': dates, 'consumption': values})

# Prepare data for LSTM model
def preprocess(df, seq_len=24):
    df['timestamp'] = pd.to_datetime(df['timestamp'])
    df.set_index('timestamp', inplace=True)
    df['hour'] = df.index.hour
    df['day_of_week'] = df.index.dayofweek
    scaler = StandardScaler()
    scaled = scaler.fit_transform(df)
    X, y = [], []
    for i in range(seq_len, len(scaled)):
        X.append(scaled[i-seq_len:i])
        y.append(scaled[i][0])  # target: consumption
    return np.array(X), np.array(y), scaler
```

```
# Build and train LSTM model
def train_model(X_train, y_train, input_shape):
    model = Sequential([
        LSTM(50, input_shape=input_shape),
        Dense(25, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
    return model

# Main
if _name_ == "_main_":
    df = generate_data()
    X, y, scaler = preprocess(df)
    split = int(len(X) * 0.7)
    X_train, X_test, y_train, y_test = X[:split], X[split:], y[:split], y[split:]

    model = train_model(X_train, y_train, (X.shape[1], X.shape[2]))
    preds = model.predict(X_test).flatten()

    # Inverse transform predictions and actuals
    def invert(y_scaled):  # Only for the first column (consumption)
        dummy = np.zeros((len(y_scaled), X.shape[2]))
        dummy[:, 0] = y_scaled
        return scaler.inverse_transform(dummy)[:, 0]

    y_test_inv = invert(y_test)
    preds_inv = invert(preds)

    # Evaluation
    print("RMSE:", np.sqrt(mean_squared_error(y_test_inv, preds_inv)))
```

```
print("MAE:", mean_absolute_error(y_test_inv, preds_inv))

# Plot
plt.figure(figsize=(10, 5))
plt.plot(y_test_inv[:200], label='Actual')
plt.plot(preds_inv[:200], label='Predicted')
plt.legend()
plt.title('Energy Consumption Forecast')
plt.show()
```

Output:



14. Future Scope

1. Real-Time Data Pipeline Integration

- **Description:**
  Incorporating a live data feed from smart meters and IoT devices would enable real-time forecasting and dynamic grid optimization.

- **Impact:**
  This would allow energy providers to respond immediately to sudden consumption spikes, improving grid reliability and customer satisfaction.

2. Advanced Visualization and Automation

- **Description:**
  Extending the dashboard using more sophisticated tools like **D3.js** for interactive web-based visualizations or **Power BI Automation** for scheduled report delivery.
- **Impact:**
  This would enhance user engagement and allow stakeholders to receive automatic updates without manual intervention, supporting faster decision-making.

3. Integration with Marketing and CRM Systems

- **Description:**
  Connecting energy consumption insights with customer relationship management (CRM) tools to enable targeted campaigns — for example, promoting energy-efficient products to high-usage households.
- **Impact:**
  This could lead to improved customer loyalty, better demand-side management, and increased adoption of green energy programs.

4. Expansion to Predictive Maintenance

- **Description:**
  Analyzing anomalies in voltage and load patterns could help predict equipment failures and schedule maintenance proactively.
- **Impact:**
  Reducing downtime and maintenance costs while extending the life of smart grid infrastructure.

15. Team Members and Roles

Name	Responsibility
VAISHNAVI V	Source code execution
VITHYASHASINI S	Exploratory data analysis
YOGALAKSHMI V	Data preprocessing