Energy Consumption Analytics for Green Power Utilities

Capstone Project Detailed Report

Presented by: Group 12

S.No.	Name	Roll Number	E-Mail ID	Contact Number
1	Shreyash Kadam	G24AI1012	g24ai1012@iitj.ac.in	+91-9172555565
2	Nitin Awasthi	G24AI1009	g24ai1007@iitj.ac.in	+91-9599089475
3	Anupkumar Pandey	G24AI1007	g24ai1009@iitj.ac.in	+91-9163310682
4	Chandan	G24AI1107	g24ai1107@iitj.ac.in	+91-9668500804
5	Vikranth Venkateswar	G24AI1083	g24ai1083@iitj.ac.in	+91-9952910682
6	Rosalin Das	G24AI1123	g24ai1123@iitj.ac.in	+91-8908850384

Team: Group 12
GitHub: [Project Repository]
Duration: 6 Weeks
Tools: Python, Pandas, NumPy, SQLite, ARIMA, Streamlit, VS Code, GitHub

1. Introduction

GreenPower Utilities, a hypothetical energy provider, seeks data-driven strategies to optimize consumption, forecasting, and anomaly detection. With smart meters, sensors, and weather APIs contributing vast time-series data, our team developed a robust data pipeline to clean, integrate, and analyze this data for actionable insights.

2. Problem Statement

- Inconsistent data from smart meters, weather APIs, and sensors.
- Challenges in aligning, cleaning, and merging heterogeneous datasets.
- Need for accurate short-term energy demand forecasting.
- Requirement for anomaly detection to preempt failures or faults.
- Visualization of insights for non-technical decision-makers.

3. Project Goals

- Create an automated, scalable ingestion pipeline.
- Standardize multi-source time-series data.
- Engineer features for effective forecasting and anomaly detection.
- Deliver interactive dashboards with operational insights.

4. Datasets Used

Weather Data: Paris (2007), sourced from OpenWeatherMap.

Energy Consumption: Smart meter logs from UCI repository.

raw_data

Characteristics:

- Irregular time intervals.
- Missing entries and sensor anomalies.
- Multiple formats and units.

5. Tools & Technologies

Languages & Libraries: Python, Pandas, NumPy, Matplotlib, Plotly

Database: SQLite (local), with TimescaleDB option

Visualization: Streamlit (planned deployment)

Version Control: Git, GitHub

Notebook Platform: Google Colab

6. Project Phases & Deliverables:

Phase 1: Data Acquisition & Cleaning:

Week 1 – Data Collection:

During this phase, we focused on collecting real-world datasets from two primary sources:

- **Energy Consumption Data**: We used the *Individual Household Electric Power Consumption* dataset from the UCI Machine Learning Repository. This dataset includes minute-level energy consumption metrics for a single household from 2006 to 2010.

- **Weather Data**: Historical hourly weather data for Paris (2007) was retrieved in the form of multiple CSV files, covering temperature, humidity, wind speed, and cloud cover, which can influence energy usage patterns.

Implementation Highlights:

- Mounted Google Drive in Google Colab to access datasets.
- Loaded the energy consumption text file using pandas.read_csv(), handling separators and missing value indicators.
- Parsed and merged the Date and Time columns to generate a unified datetime index.

- Loaded all weather data CSVs from the specified folder using glob and concatenated them into a single DataFrame.

Week 2 – Data Cleaning and Preprocessing:

This phase was dedicated to cleaning and preparing the data for modeling and visualization.

Energy Data Cleaning:

- Converted all relevant columns to numeric, coercing non-numeric entries to NaN.
- Dropped rows with missing values.
- Filtered data to keep only records from the year 2007.
- Removed outliers from Global_active_power using the Interquartile Range (IQR) method.
- Selected only numerical features for processing.
- Resampled the data to hourly intervals to reduce noise and ensure consistency with weather data.

Weather Data Cleaning:

- Parsed datetime strings and dropped invalid rows.
- Set the datetime column as the index and selected relevant columns: temp, humidity, windspeed, cloudcover.
- Applied time-based interpolation to fill in missing values.
- Sorted index and filtered the weather data to 2007.

Data Integration:

- Merged the cleaned energy and weather datasets on the hourly datetime index using an inner join.
- This resulted in a clean, merged dataset with 11 continuous variables and a total of 7357 hourly entries for 2007.
- Saved the combined dataset as a CSV for downstream analysis.

Phase 2: Ingestion Pipeline & Feature Engineering:

Week 3 – Data Ingestion Pipeline and Storage:

This week focused on implementing an automated pipeline to ingest cleaned, preprocessed time-series data into a time-series optimized storage system — in this case, SQLite (with potential extension to TimescaleDB).

Objectives Achieved:

- Load hourly-level energy and weather data into a structured **SQLite database**.
- Design and execute date-partitioned ingestion to handle large time-series datasets efficiently.
- Avoid redundant inserts by checking existing records before insertion.
- Preserve data schema for consistent querying and further transformations.

Step 1: Prepare Clean Data

- The cleaned_combined_data_2007.csv file generated in Phase 1 was loaded into a DataFrame.
- Parsed the datetime column and set it as the index.
- All column names were normalized to lowercase to maintain uniformity in the database.

Step 2: Create SQLite Database and Schema

- A local SQLite database (energy_timeseries.db) was initialized.
- A table timeseries_energy_weather was created with 10 key features (excluding reactive power and intensity to reduce redundancy).

Step 3: Batch-wise Daily Ingestion

- The ingestion function insert_daily_chunk() filters daily data and inserts only non-duplicate rows into the database.
- Date range: 2007-01-01 to 2007-12-31 (365 days).
- Insertion was done one row at a time to prevent constraint violations on duplicate primary keys.

Validation

- Ran test SQL queries to:
    - Verify sample entries.
    - Count total rows (expected ~7356).
    - Confirm schema correctness.

Final Output

The database was saved to Google Drive for persistence and further querying:

Why SQLite?

- Lightweight, fast, and ideal for local development.
- Schema enforcement and SQL queries make it easy to test before scaling to platforms like TimescaleDB or PostgreSQL.

Week 4 – Feature Engineering:

In Week 4, we focused on transforming the ingested data into **analytical features** that can improve model performance and support operational decisions. These features were generated at both **daily and hourly levels**, enabling a flexible basis for demand forecasting, anomaly detection, and usage pattern analysis.

Objectives Achieved:

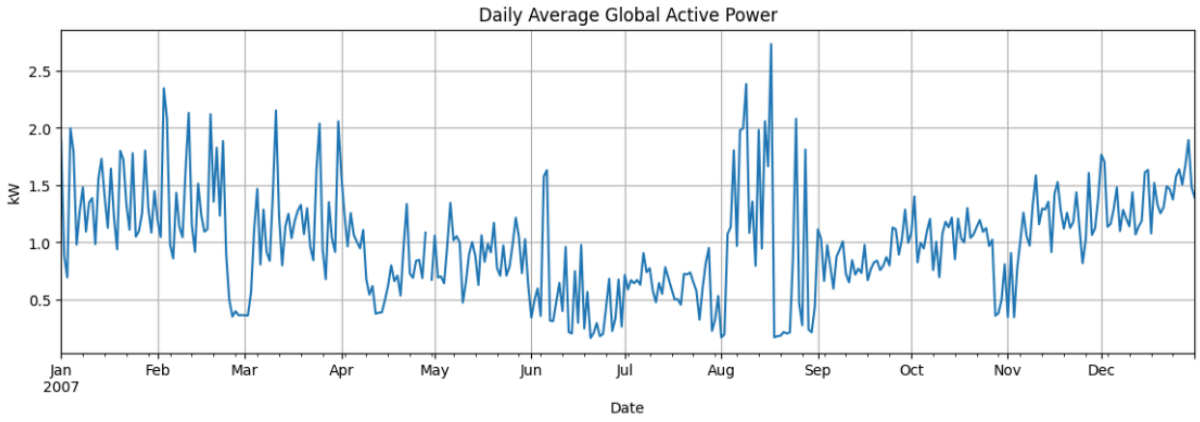
- Create time-based aggregation features (mean, max, min).
- Identify consumption patterns based on hour-of-day.
- Quantify peak load behaviors.
- Correlate weather with energy consumption.
- Prepare feature tables for modeling.

1. Daily Aggregation

To understand consumption trends at a macro level:

- **Daily averages** of global_active_power were calculated using .resample('D').mean().
- This helps in identifying weekday-weekend patterns and seasonal variation.

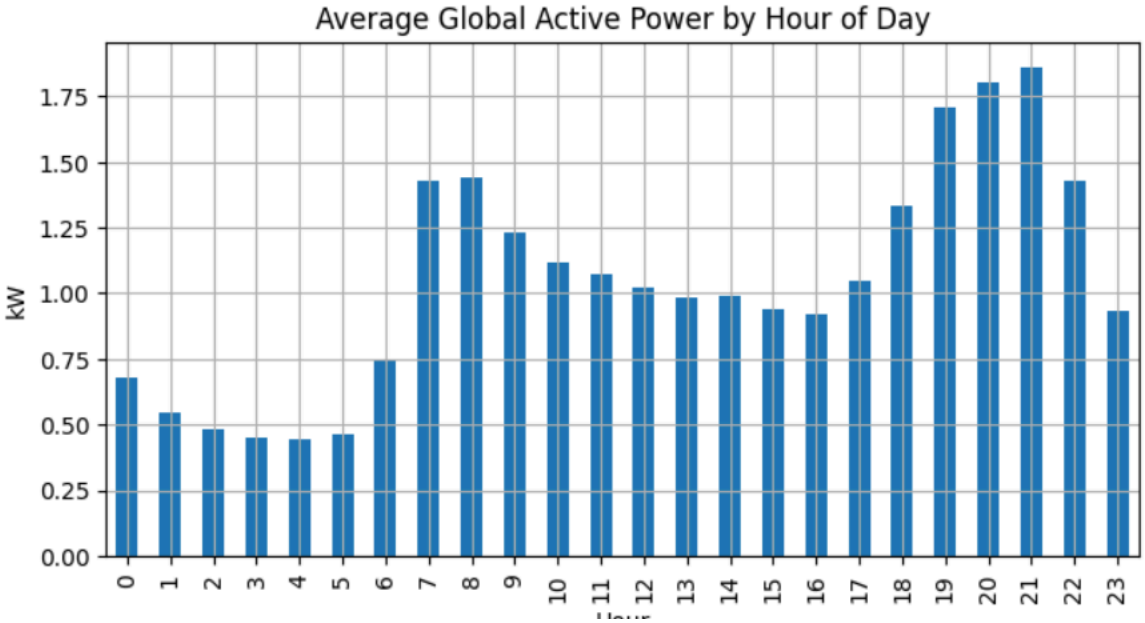
Line plot of daily average consumption.



2. Hourly Consumption Patterns

- Grouped data by hour (0–23) to compute **hourly average power consumption**.
- Helps identify **peak usage hours**, which are crucial for load balancing and scheduling.

Bar plot of hourly average consumption.

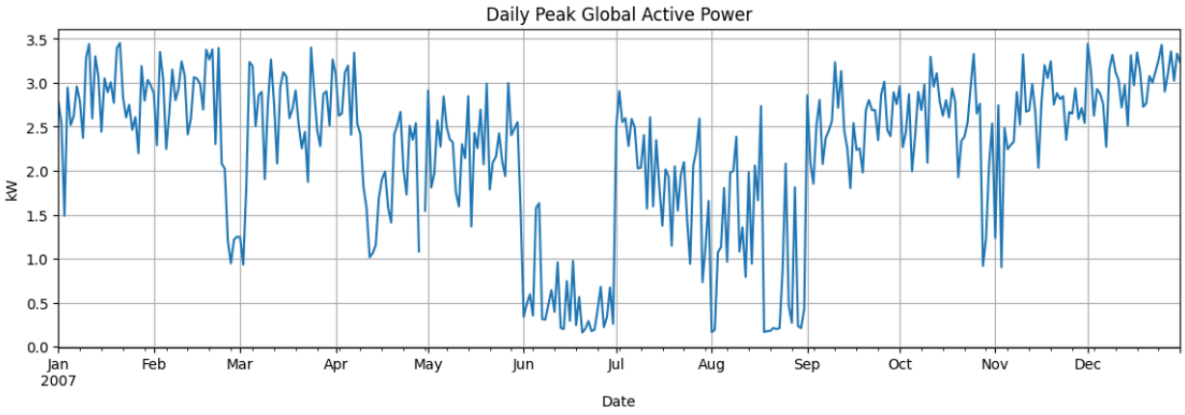


Insights:
Evening hours (19:00–21:00) showed the highest average consumption (~1.8 kW), consistent with typical residential usage patterns.

3. Peak Load Statistics

- Computed **overall peak** power and timestamp using .max() and .idxmax().
- Also calculated **daily peak load values** for modeling peak demand scenarios.

Line plot of daily peak loads.



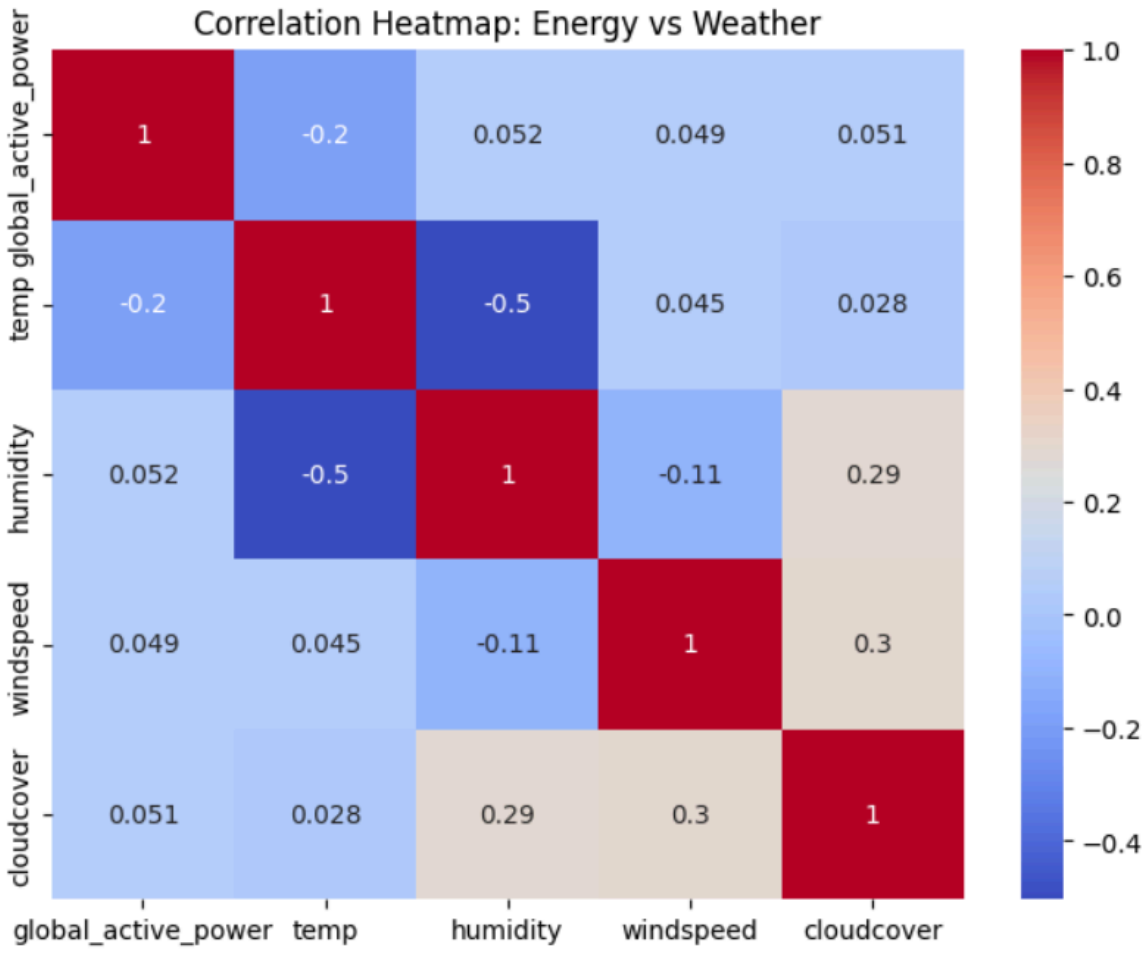
Insight:

Peak load occurred at **19:00 on Jan 21, 2007** with a power draw of **3.45 kW**.

4. Weather Correlation Analysis

- Generated a **correlation matrix** between energy consumption and weather variables.
- Visualized results using a **Seaborn heatmap**.

Heatmap for correlation between weather and power usage.



Key Findings:

- Weak negative correlation between **temperature** and consumption (-0.19) suggests slightly higher energy use on colder days (likely heating).
- Other weather variables showed minimal influence.

5. Database Indexing

- Created an index on the datetime column in SQLite to **speed up time-based queries** and improve performance during data access for dashboards or modeling.

6. Feature Table Creation

Two primary feature tables were created for downstream use:

Feature_table_daily: Daily mean, max, min of power; mean temp & humidity

Feature_table_hourly: Hourly mean of power, temp, humidity

Phase 3: Forecasting & Anomaly Detection:

Week 5: Forecasting & Anomaly Detection:

In Week 5, the focus shifted toward building a **time-series forecasting model** to predict daily energy consumption and applying **anomaly detection techniques** to identify irregularities that may indicate outages or operational faults.

Objectives Achieved:

- Build a statistical ARIMA model to forecast energy demand.
- Validate predictions using real consumption data.
- Detect sudden drops in usage, indicating potential anomalies or failures.
- Visualize predictions and anomalies.
- Export forecast results for dashboard integration.

1. Forecasting with ARIMA

Time-Series Preparation

- Loaded cleaned hourly dataset and resampled to daily average consumption (Global_active_power).
- Ensured time-series continuity and handled any missing values.

Stationarity Check

- Performed **Augmented Dickey-Fuller (ADF)** test.
- ADF Statistic: -3.46, **p-value: 0.0089** → Indicates **stationary series**, so differencing was not needed.

Train-Test Split & Model Building

- Split data:
    - **Training**: Jan to Nov 2007
    - **Testing**: Dec 2007
- Built an **ARIMA(6,1,5)** model (manually tuned for best performance).
- Used Statsmodels' ARIMA class to fit the model.

Forecasting & Visualization

- Forecasted Dec 2007 (test period) and compared with actuals.
- Generated **30-day future forecast** beyond available data (Jan 2008).
- Visualized training, test, and forecast curves.

2. Anomaly Detection:

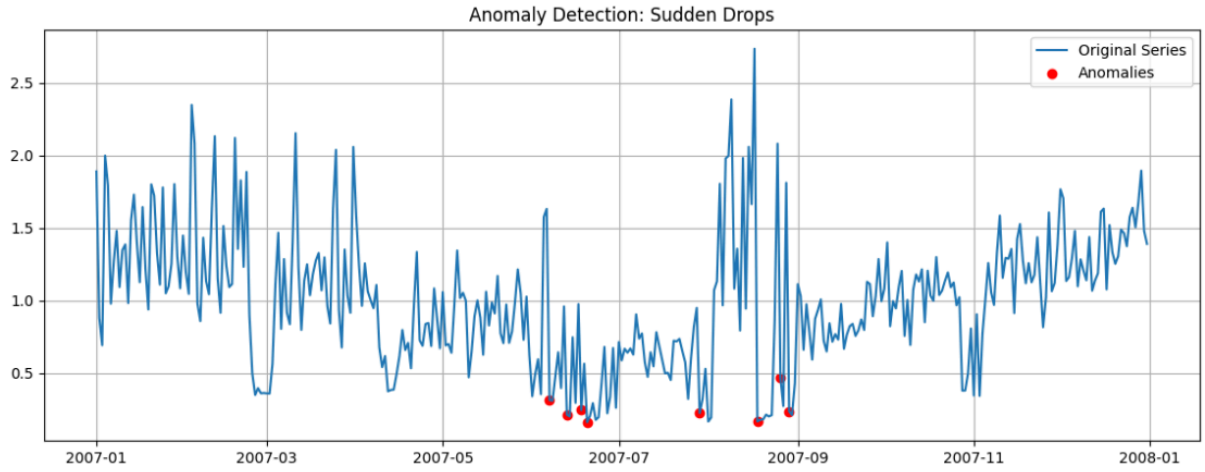
To detect anomalies like sudden drops in energy usage, we used a thresholding technique:

Method:

- Computed 1-day lag and difference.
- Flagged data points where the difference was less than -70% of the previous day's value.

These events likely indicate **sensor failure**, **data gaps**, or **outages**.

Visual: Overlayed anomalies as red points on the daily energy consumption graph.



Key Learnings This Week

- Time-series modeling with ARIMA requires careful differencing and hyperparameter tuning.
- Forecasts must be validated with metrics (MSE, MAE) and visualization.
- Anomaly detection complements forecasting by identifying operational outliers.

Week 6: Visualization Dashboard & Final Recommendations:

In the final phase of the capstone project, we developed an interactive dashboard using Streamlit and Plotly to present our forecasting results and draw actionable recommendations from energy consumption trends. This week marked the transition from backend analytics to frontend insights delivery.

Objectives Achieved:

- Deploy an interactive, filterable dashboard for stakeholders.
- Visualize forecast vs actual consumption with clear date-based filtering.
- Enable monitoring of consumption trends.
- Provide recommendations for operations and forecasting refinement.

Streamlit Dashboard Components

1. Data Loading

- The arima_model_forecast.csv file, which contains actual and forecasted consumption, was loaded with date parsing.

2. Sidebar Date Filter

- A dynamic date filter was added to the sidebar, allowing users to interactively filter results based on any time window from Jan 2007 to Feb 2008.

3. Forecast vs Actual Plot

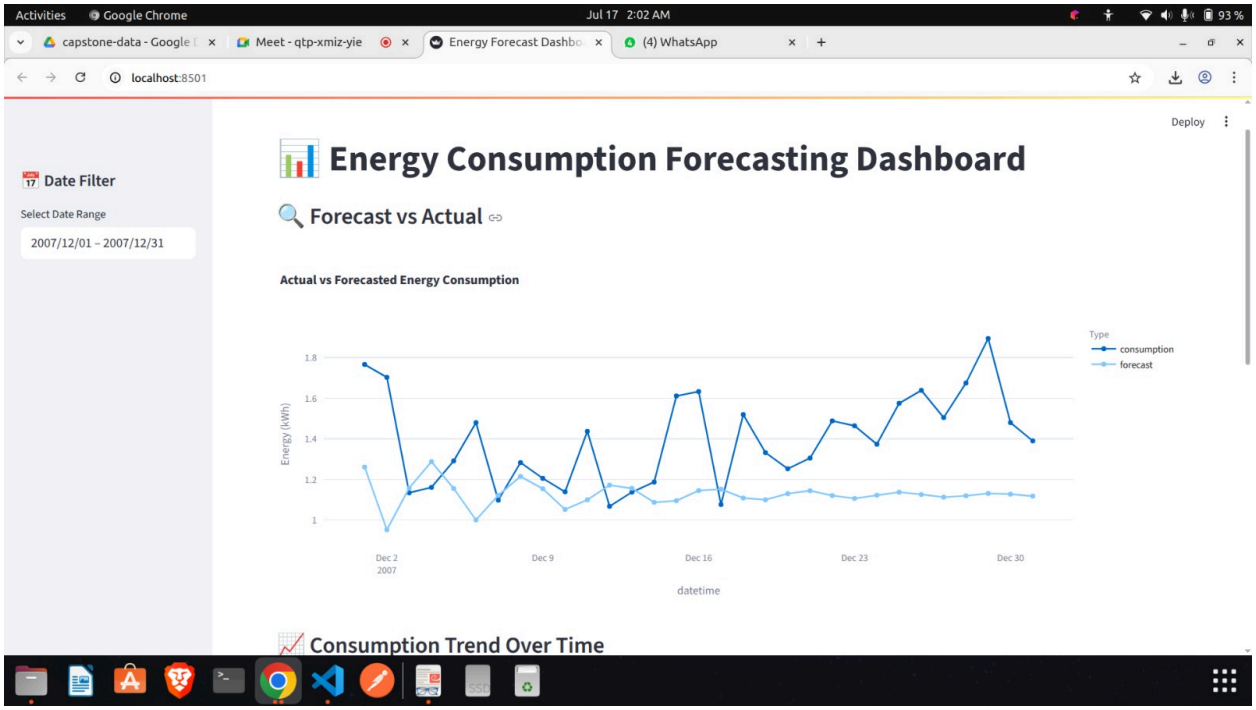
- A **multi-line Plotly chart** compares actual vs forecasted energy consumption.
- Hover, zoom, and tooltip interactivity supports detailed analysis by users.

4. Consumption Trend Plot

- A dedicated line chart for just actual consumption helps track seasonality and operational spikes.

5. Recommendations Section

Clear and actionable strategies were summarized based on model insights and consumption behavior:



Deployment Benefits

- **Real-time Filtering:** Allows analysts to monitor specific date ranges.
- **Scalable UI:** Can easily integrate additional features (e.g., anomaly alerts, heatmaps).
- **User-Friendly Visuals:** Enables operational teams to make data-driven decisions quickly.

7. Visualization & Dashboards:

- Time-series plots of energy usage.
- Forecast vs. actual consumption overlays.
- Outlier and anomaly visualizations.
- Planned deployment of dashboards using **Streamlit**.

8. Challenges Faced

- Overlapping timestamps and irregular intervals.
- Complex merging of weather and consumption data.
- Tuning forecasting models to handle daily/seasonal variance.
- Resolving unit mismatches and missing metadata.

9. Learnings & Outcomes

- Hands-on with real-world time-series cleaning and alignment.
- Exposure to end-to-end data pipeline development.

- Forecasting using classical models and performance evaluation.
- Real-time dashboard planning for operational insight delivery.

10. Conclusion

Our project successfully showcases how modern data engineering and analytics techniques can be applied to solve pressing challenges in the energy sector. Through effective data integration, forecasting, and visualization, we've provided a blueprint that can enhance utility operations while supporting sustainability.