PROJECT TITLE

Power Consumption Analysis

For Household

Artificial Intelligence & Machine Learning

**Team ID:** LTVIP2025TMID49661

**Team Size:** 4

**Team Leader:** Sirigiri Kiran(228X1A4541)

**Team member:** Srihari Narisetty(228X1A4529)

**Team member:** Gorantla Hemanth(228X1A4511)

**Team member:** Thammisetty Rakesh(228X1A452)

## Table of Contents

1.Introduction

2.Project Overview

3.System Architecture

4.Setup Instructions

5.Folder Structure

6.Running The Application

7.API Documentation

8.User Interface (Screenshot section)

9.Testing

10.Demo Video

11.Known Issues

12.Future Enhancements

1.INTRODUCTION

Welcome to the **"** **Smart Power Consumption Analyzer "** also known as **"** **Power Consumption Analysis for Households"!** This project is designed to address a common challenge faced by modern households: effectively monitoring and analyzing electricity usage to promote energy efficiency and reduce costs.

At its core, this application harnesses the power of **Machine Learning** to deliver intelligent insights into energy consumption patterns. By training models on real-world household energy usage data, we can accurately identify trends, predict future consumption, and detect unusual spikes that may indicate inefficiencies or faulty appliances.

We leverage powerful algorithms such as **Linear Regression, LSTM networks** to analyze both historical and real-time energy data. These models learn from various inputs—like time of day, seasonality, and appliance usage—to make predictions and provide recommendations tailored to individual household habits.

The system is deployed as an intuitive **Flask web application**, making it accessible through a simple browser interface. Users can upload their energy consumption data (from smart meters or logs), and the backend, powered by our trained ML models, will process the data and return insights—such as "High usage detected during peak hours," "Estimated bill for next month," or "Possible energy savings by shifting usage to off-peak times."

This project aims not only to provide a practical tool for energy monitoring and optimization but also to serve as a valuable educational resource—covering everything from data collection and preprocessing to machine learning model development and full-stack web deployment.

2.PROJECT OVERVIEW

The **"Smart Power Consumption Analyzer"** is an innovative full-stack web application designed to transform how households understand and manage their electricity usage. Its core mission is to analyze, predict, and visualize energy consumption patterns using advanced machine learning techniques, empowering users to make informed decisions to reduce waste and optimize usage.

This project addresses a key challenge in energy management: the lack of intelligent, accessible tools for consumers to monitor their electricity usage in real time. Many households either rely on manual tracking or receive only high-level monthly summaries, which limits their ability to respond to spikes, inefficiencies, or abnormal usage.

The Smart Power Consumption Analyzer offers an automated, insightful, and interactive solution that can help households reduce energy bills, improve appliance efficiency, and support sustainable living.

**PURPOSE:**

The primary objective of this project is to demonstrate a practical application of machine learning and web technologies in addressing real-world energy efficiency challenges. Specifically, the application aims to:

* **Monitor and Analyze Usage**: Track household power consumption over time and break it down by time of day, appliance, or user behavior.
* **Predict Future Consumption**: Use trained models to forecast electricity usage for upcoming hours, days, or months.
* **Reduce Wastage**: Help users identify patterns of overuse or inefficiency, allowing for corrective action.
* **Promote Energy Awareness**: Encourage smarter usage habits by providing visual feedback and intelligent recommendations.

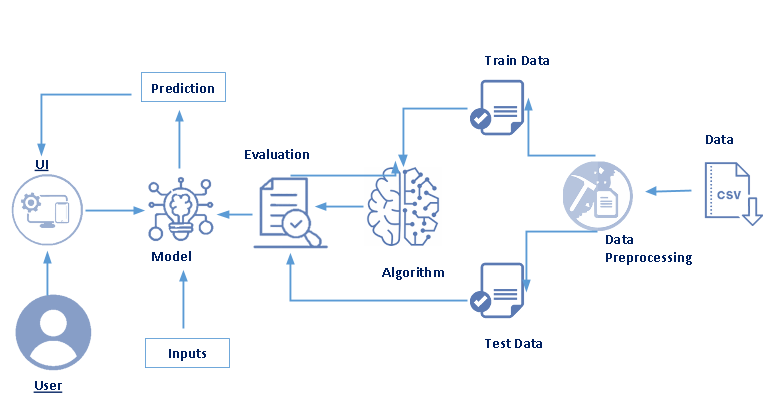
**FEATURES:**

The **Smart Power Consumption Analyzer** is built with both usability and technical robustness in mind, offering the following key features:

* **User-Friendly Web Interface**: An intuitive dashboard allows users to upload CSV datasets (from smart meters or manual logs) or connect to live data streams.
* **Real-time Analysis and Predictions**: Instantly visualizes trends and provides consumption predictions using trained machine learning models.
* **Machine Learning Core**: The backend leverages models like Linear Regression and LSTM (for time-series data), trained on historical power usage to forecast future trends.
* **Insightful Visualizations**: Interactive graphs (via libraries like seaborn) show daily/weekly usage patterns, peak hours, and appliance-specific consumption.
* **Scalable Architecture**: Built with Python (Flask) on the backend and a responsive frontend, the system is designed for future enhancements such as real-time IoT integration or multi-user support.

This project serves as a full-stack demonstration—from data ingestion and preprocessing to model deployment and frontend visualization—of how AI and machine learning can be used to empower more sustainable and cost-effective household energy management.

3.ARCHITECTURE

 **Frontend**

The **Frontend** acts as the user interface layer—the bridge between the user and the machine learning system. According to the image, the frontend is responsible for:

**Key Components:**

* **UI (User Interface)**:
  + A web-based interface built with **HTML**, **CSS**
  + Allows the user to:
    - Upload power consumption datasets (typically in .csv format).
    - View predictions, trends, and analytics.
    - Receive feedback from the backend (e.g., forecasted usage, cost estimation, alerts).
* **User Interaction Flow**:
  + The user uploads a .csv file or provides live inputs (e.g., from a smart meter).
  + These inputs are sent to the backend via POST requests.
  + Prediction results are rendered visually (graphs, tables, messages).

**Technologies (Suggested):**

* HTML5 + CSS3 + Bootstrap (for styling and layout)
* Seaborn (for graphs and data visualizations)

**Backend**

The **Backend** handles all the machine learning logic, data processing, and model serving. Based on the flow in the image, the backend includes several critical stages:

**Key Components:**

1. **Data Preprocessing**:
   * Raw data (CSV) is cleaned, normalized, and transformed into a suitable format.
   * Missing values, time-based aggregations, and feature extraction are handled here.
2. **Train/Test Split**:
   * Preprocessed data is divided into **Train Data** and **Test Data** for training and evaluating the model.
3. **Algorithm / Model Training**:
   * Machine learning algorithms (e.g., **LSTM**,**Linear Regression**) are trained using the training dataset.
   * Evaluation is done on the test set to validate model performance.
4. **Model Deployment**:
   * A trained model is saved (e.g., model.pkl) and loaded by the Flask app.
   * When a user uploads new data, the model processes it and returns predictions.
5. **Prediction & Evaluation**:
   * The model outputs usage predictions or alerts (e.g., expected peak hours or overconsumption).
   * Evaluation metrics (e.g., MAE, RMSE, R²) can also be returned for transparency.

**Technologies (Suggested):**

* **Python** for core backend logic
* **Flask** or **FastAPI** as the web framework
* **Pandas**, **NumPy** for data processing
* **Scikit-learn** for ML models
* **Matplotlib**, **Seaborn**, or **Plotly** for backend-generated graphs

4.SETUP INSTRUCTIONS

**Prerequisites**

Before running the project, ensure you have the following installed and configured:

* **Python 3.x**
* **Pip** (Python package installer)
* **Required Python libraries**, such as:
  + Flask (for web application backend)
  + Pandas, NumPy (for data handling and numerical computations)
  + Matplotlib, Seaborn, or Plotly (for visualizations)
  + Scikit-learn,(for machine learning and time-series forecasting)
* **Google Colab** (optional, recommended for training ML models with GPU acceleration)
* **Ngrok** (for exposing your local or Colab Flask app to the internet)

**Installation**

**1. Clone the Repository**

git clone https://github.com/yourusername/PowerConsumptionAnalyzer.git

**2. Navigate into the Project Directory**

cd PowerConsumptionAnalyzer

**3. Install Python Dependencies**

You can install the required libraries using pip:

pip install Flask pandas numpy matplotlib scikit-learn tensorflow

Alternatively, use the requirements.txt file (if available):

pip install -r requirements.txt

**4. Download the Dataset and Pre-trained Model**

* If a **pre-trained model** file (e.g., power\_forecast\_model.h5) is included in the repository:
  + Ensure you have **Git LFS** installed and run:
  + git lfs pull
* The **dataset** (for retraining or testing) can be obtained from public sources like the UCI Machine Learning Repository or Kaggle.
* Save the dataset (usually a .csv file) in the project’s data/ folder or the root directory.

5.FOLDER STRUCTURE

### Client (Frontend):

**templates/:** Contains HTML files that define the user interface (e.g., index.html,result.html).

**static/:** (Expected if present) Would contain static assets like css/ for stylesheets for client-side and uploads/ for temporary uploaded images.

### Server (Backend):

**app.py:** The main Flask application file, handling routes, model loading, takin values, and predictions.

**PCA.ipynb: jupyter notebook** Script for training and evaluating the machine learning model.

**data\_prep\_and\_viz.py:** Script for data preprocessing and visualization.

**Model.pkl:** The pre-trained jupyter notebook saved model file used for predictions.

6.RUNNING THE APPLICATION

**Local Setup:**

1. Make sure Python and all dependencies are installed.
2. Launch the Flask backend:

python app.py

1. python app.py
2. Open your browser and go to:  
   <http://127.0.0.1:5000>

**Google Colab Setup:**

1. Mount your Google Drive and unzip the project into /content/SmartPowerAnalyzer/
2. Run the setup and app scripts inside Colab
3. Use ngrok to expose the app:

!ngrok http 5000

4. Access the public URL from ngrok to use the app in real time

7.API DOCUMENTATION

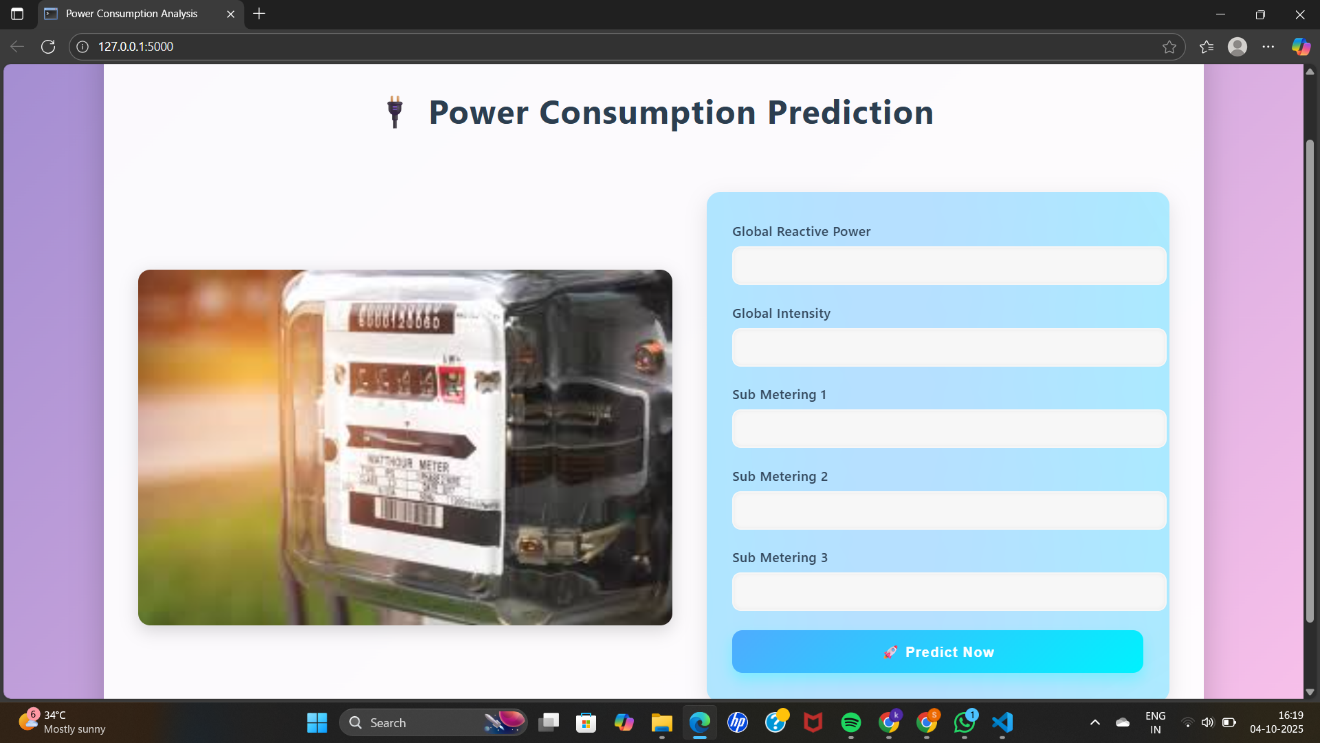
**7. API Documentation**

While the application is primarily web-based, the backend exposes a few useful endpoints:

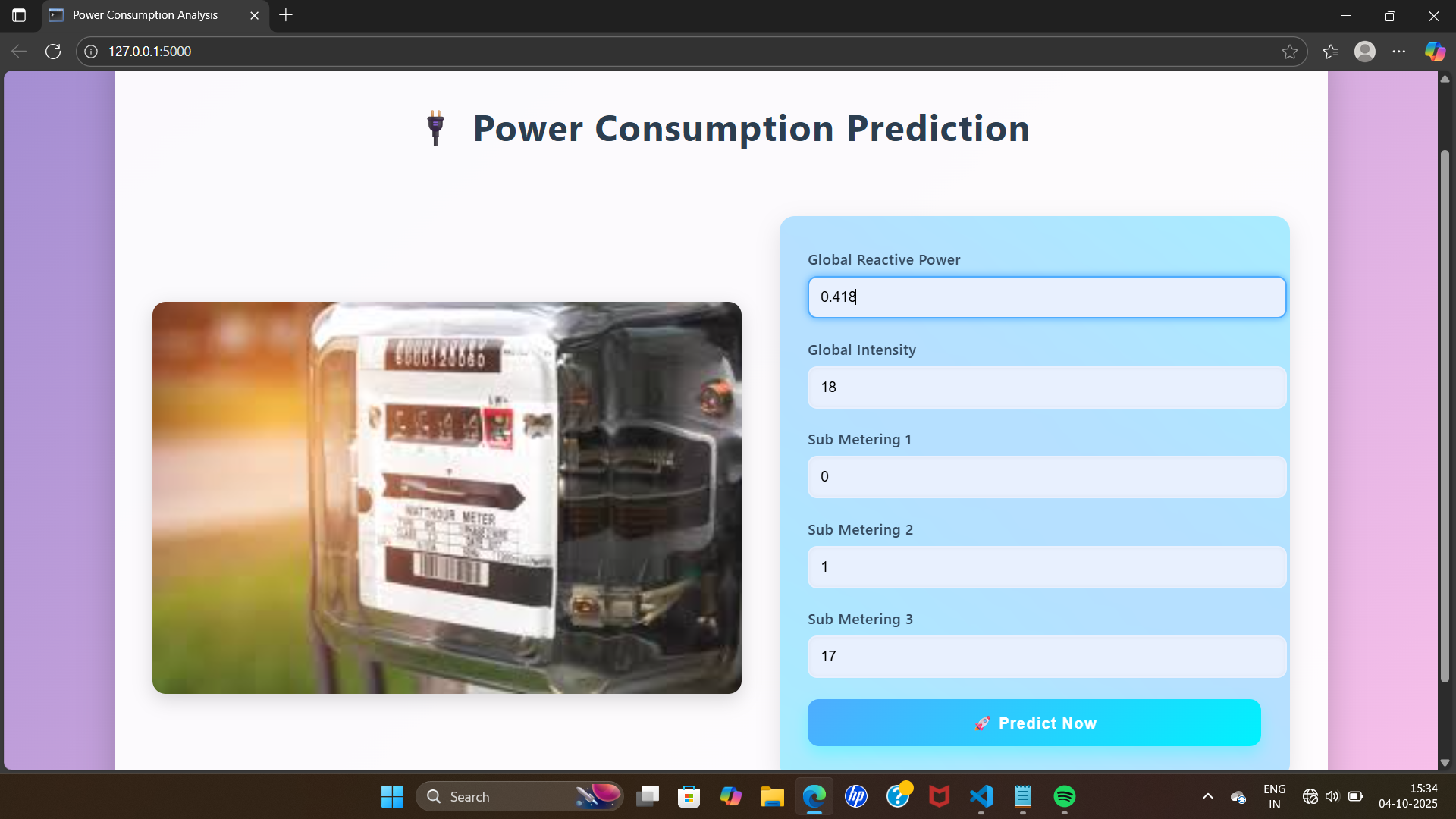
* / (GET): Loads the main dashboard and upload interface.
* /analyze (POST): Accepts CSV data of power consumption and returns usage analytics and predictions.
* /visualize (GET): Generates graphs and summaries based on recent uploads.

8.USER INTERFACE(SREENSHOT SECTION)

The UI is a simple web page allowing users to upload an image file. After processing, it displays the prediction result clearly

**THE WEPAGE ALLOWING USER TO GIVE THE INPUTS: **

**PREDICTION RESULTS:**

****

****

9.TESTING

**Model Testing**: Accuracy, RMSE, and MAE are evaluated during model training (see model\_training.py).

**Web App Testing**: Manual and unit testing of Flask routes, file upload handling, and edge-case data inputs.

**Tools**: pytest, unittest, and manual validation with sample datasets

10.DEMO VIDEO

11.KNOWN ISSUES

**Data Format Sensitivity**: The app expects specific CSV column names and formats (e.g., time-series with "timestamp" and "power" columns).

**Model Generalization**: Forecast accuracy may drop for households with unusual or unpredictable usage patterns not represented in training data.

**Scalability**: Currently optimized for small-scale usage. Production deployment would require load balancing, Dockerization, and cloud services.

12.FUTURE ENHANCEMENTS

* **Live Data Integration**: Connect directly to smart meters or IoT devices for real-time monitoring.
* **User Accounts**: Implement login and personalized dashboards with consumption history.
* **Mobile App Version**: Develop a mobile-responsive version or standalone app for Android/iOS.
* **Automated Alerts**: Notify users via email/SMS when usage exceeds thresholds.
* **Cloud Deployment**: Deploy using AWS, GCP, or Azure with autoscaling and secure data handling.
* **Dynamic Appliance Detection**: Use machine learning to identify which appliance is consuming how much power based on usage patterns.