**Energy Insight Predictor**

**CS19643 – FOUNDATIONS OF MACHINE LEARNING**

Submitted by

KAMAL J R 220701117

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE**

**ANNA UNIVERSITY, CHENNAI**

**MAY 2025**

**BONAFIDE CERTIFICATE**

Certified that this Project titled **“ENERGY INSIGHT PREDICTOR”** is the bonafide work of **“KAMAL J R (2116220701117)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

**Dr. V. Auxilia Osvin Nancy., M.Tech., Ph.D.,**

SUPERVISOR,

Assistant Professor

Department of Computer Science and Engineering,  
Rajalakshmi Engineering College, Chennai - 602 105.

Submitted to Mini Project Viva-Voce Examination held on

**Internal Examiner External Examiner**

**ABSTRACT**

With the era of fast urbanization and the increasing need for sustainable energy consumption, accurate prediction of household power consumption has become an essential aspect of smart grid development and energy management systems. This mini project aims to develop a machine learning-based predictive model to forecast household global active power consumption using historical time-series data. The project aims specifically at the application of the Random Forest Regression algorithm to analyze and forecast energy consumption based on a range of temporal and contextual features derived from the dataset. The dataset utilized is the Individual Household Electric Power Consumption dataset, which includes over two million readings of power consumption, recorded in one-minute intervals over nearly four years. These readings include a range of electrical parameters such as global active power, voltage, global reactive power, and energy sub-metering values.

The project pipeline starts with the integration and preprocessing of raw data through Python in the Google Colab environment. The dataset, initially in.txt format and semicolon-delimited, is parsed and cleaned by merging the 'Date' and 'Time' columns into one datetime object. This allows temporal feature extraction, including hour of the day, day of the week, month, and year, which are crucial in capturing periodic and seasonal patterns in energy consumption. Comprehensive data cleaning procedures are executed to address missing values by forward and backward filling, and all numerical values are strictly converted from strings to appropriate numeric types to maintain analytical consistency. Exploratory Data Analysis (EDA) provides us with some interesting insights into the patterns of energy consumption, like peak consumption during certain hours of the day and different consumption patterns during different months. The interdependence of the numerical features is analyzed using a correlation heatmap, which reveals the most powerful predictors. The data is divided into the training set and the testing set based on an 80-20 ratio, and the features are normalized using standard scaling for improved model performance.

Random Forest Regressor, a robust ensemble learning algorithm, is used for modeling. Preprocessed data is used to train the model, and the model is tested on the basis of metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the R-squared (R²) score. The model demonstrates strong prediction accuracy in the train and test sets, indicating its ability to learn nonlinear complex relationships within the dataset. Additionally, a feature importance plot is generated to observe which temporal and electrical features significantly affect energy consumption, and 'hour of the day' and 'sub-metering' features are some of the contributing top features.

The project is finalized and validated with some visual diagnostics and checks like actual vs. predicted power plots and residual error plots to verify model bias and variance. The model is saved and exported using joblib for future reuse in real-time applications or additional fine-tuning. This work not only provides an efficient data-driven solution for energy forecasting but also opens the door to more intelligent demand-side energy management systems. Future work could involve hyperparameter optimization, addition of external variables like weather data, or a more advanced transition to the use of more advanced deep learning methods for improved temporal accuracy and scalability.

**ACKNOWLEDGMENT**

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Dr. V. AUXILIA OSVIN NANCY., M.Tech., Ph.D.,** Professor of the Department of Computer Science and Engineering. Rajalakshmi Engineering College for his valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Mrs. V. AUXILIA OSVIN NANCY** Assistant Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

KAMAL J R – 2116220701117

**TABLE OF CONTENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO.** | | **TITLE** | **PAGE NO.** |
|  | | **ABSTRACT** | **1** |
|  | | **ACKNOWLEDGEMENT** | **2** |
|  | | **LIST OF TABLES** | **4** |
|  | | **LIST OF FIGURES** | **5** |
| **1.** | | **LIST OF ABBREVIATION**  **INTRODUCTION**   * 1. GENERAL | **6**  **7** |
| **2.**  **3.** | | * 1. OBJECTIVE   2. EXISTING SYSTEM   **LITERATURE SURVEY**  **PROPOSED SYSTEM** | **9**  **10** |
|  |  | 3.1 GENERAL |  |
|  | | 3.2 SYSTEM ARCHITECTURE DIAGRAM  3.3 DEVELOPMENT ENVIRONMENT |  |
|  | | 3.3.1 HARDWARE REQUIREMENT  3.3.2 SOFTWARE REQUIREMENT  3.4 DESIGN THE ENTIRE SYSTEM  3.4.1 ACTIVITY DIAGRAM  3.4.2 DATAFLOW DIAGRAM  3.5 STATSTICAL ANALYSIS  **MODULE DESCRIPTION**  4.1 SYSTEM ARCHITECTURE  4.1.1 USER INTERFACE DESIGN  4.1.2 BACKEND INFRASTRUCTURE  4.2 DATA COLLECTION & PREPROCESSING  4.2.1 DATA VALIDATION  4.2.2 DATA FORMATTING  4.2.3 ERROR HANDLING |  |
| **5.**  **6.** | | 4.3 SYSTEM WORKFLOW  4.3.1 USER AUTHENTICATION  4.3.2 DATA COLLECTION  4.3.3 MODEL TRAINING  4.3.4 PREDICTION  4.3.5 DATA LOGGING  4.3.6 USER LOGOUT  **IMPLEMENTATION** 5.1 IMPLEMENTATION  5.2 RESULT  5.3 OUTPUT AND SCREENSHOTS  **CONCLUSION AND FUTURE ENHANCEMENT** 6.1 CONCLUSIONS  6.2 FUTURE ENHANCEMENTS | **29**  **32** |
| **7.** | | **REFERENCES** | **34** |

**4.**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| FIGURE NO. | TITLE | PAGE NO. |
| 3.1 | Architecture diagram | 14 |
| 3.2 | Activity Diagram | 16 |
| 3.3 | Context Level DFD | 17 |
| 3.4 | Level-1 DFD | 18 |
| 3.5 | Level-2 DFD | 19 |
| 5.1 | Import libraries and dependencies | 29 |
| 5.2 | Import data and read the file | 29 |
| 5.3 | Plot Global Active Power Vs Date | 30 |
| 5.4 | Correlation Matrix of data | 30 |
| 5.5 | Average Global Active Power by Hour of Day | 31 |
| 5.6 | Actual Vs Predicted Global Active Power | 31 |
| 5.7 | Residual Plot | 32 |
| 5.8 | Model built and saved | 32 |
| 5.9 | Metrics | 32 |
| 5.10 | Predicted Output as Result | 33 |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| SNO | ABBR | EXPANSION |
| 1 | MSE | Mean Squared error |
| 2 | AI | Artificial Intelligence |
| 3 | ANN | Artificial Neural Network |
| 4 | API | Application Programming Interface |
| 5 | CSV | Comma-Separated Values |
| 6 | CPU | Central Processing Unit |
| 7 | DL | Deep Learning |
| 8 | EDA | Exploratory Data Analysis |
| 9 | GUI | Graphical User Interface |
| 10 | IoT | Internet of Things |

# CHAPTER 1

**1.INTRODUCTION**

* 1. **GENERAL**

Global energy consumption is increasing exponentially, particularly in domestic and commercial markets. As households increasingly depend on electrical appliances and smart devices, learning and predicting electricity usage becomes increasingly important. Efficient energy forecasting not only saves energy and money but is also of utmost importance in power system planning, load balancing, and power failure prevention. To this end, machine learning has emerged as a powerful instrument in energy consumption prediction since it is capable of handling large volumes of data and extracting complex, non-linear relationships that may elude traditional methods.

This "Prediction of Household Global Active Power Consumption using Random Forest Regression" project intends to predict household energy consumption via machine learning, the Random Forest algorithm. The data source of this research, which was drawn from the "Individual Household Electric Power Consumption" database, comprises over 2 million readings taken over nearly four years. The readings are for various electrical attributes such as global active power, global reactive power, voltage, current, and sub-metered energy consumption for various areas of a home. The project is executed using the Google Colab platform with the help of Python with libraries such as Pandas, NumPy, Scikit-learn, and Matplotlib. The project begins with data preprocessing to handle missing values, date and time format conversion, and feature extraction from the data. After preprocessing, data visualization techniques are used to detect hidden patterns, correlation, and anomalies. The data is then split into training and test sets, and a Random Forest Regressor is used to predict future energy consumption. The model performance is evaluated based on industry parameters such as MAE (Mean Absolute Error), MSE (Mean Squared Error), RMSE (Root Mean Squared Error), and R² score.

This system can empower policymakers, energy providers, and residents with predictive data on consumption patterns that can inform the effort to optimize energy distribution, create demand-response strategies, and improve energy-saving strategies. The project also encourages a data-driven solution to real-world issues through machine learning and hence to sustainable living and technological innovation.

* 1. **OBJECTIVE**

The primary goal of this project is to train and deploy a machine learning model which can predict world active power consumption at a residence from historical data. The primary goals of the system are. In a bid to pre-process and clean raw energy consumption data to give analytical consistency and reliability. To extract and engineer features from time data (hour, day, month) in order to recognize trends and cyclic patterns in power consumption. In order to create a robust Random Forest Regression model that can deal with complex patterns of electricity consumption. To compare model performance using standard regression metrics and test its stability on new data. To plot real vs. estimated energy consumption using graphs for easier interpretation of model results. To provide insights into which features (e.g., hour of day, voltage, sub-metering data) most influence consumption levels. To export and save the trained model for future use in real-time or deployment. All these goals seek to prove the worth of machine learning in energy prediction and how it can be utilized in actual domestic energy management systems.

* 1. **EXISTING SYSTEM**

Current prediction methods for electricity use in residential environments are largely based on conventional statistical models like ARIMA (Auto-Regressive Integrated Moving Average), linear regression, or basic rule-based systems. Although these are appropriate for linear and stationary data, they are not very effective with highly volatile and non-linear data, which are typical in actual household consumption patterns. Furthermore, they are not necessarily effective in dealing with missing values and tend to make strong assumptions about data distribution and stationarity.

In nearly every residential setup, there is no predictive system at all. Homeowners themselves are not aware of their usage patterns until they receive monthly bills. The passive methodology does not permit them to make proactive actions like modifying usage at peak times, purchasing energy-efficient appliances, or enrolling in smart grid initiatives. Energy monitoring practices like checking meter readings or looking at past utility bills are non-real-time and do not have the amount of detail needed for useful analysis.

Furthermore, conventional systems seldom include external or contextual information such as the time of day, day of the week, or sub-metering breakdowns, which are important for optimized forecasts. These systems also do not have the flexibility and learning ability of machine learning models, i.e., they do not get better over time or learn from emerging patterns in data.

Conversely, the system herein utilizes a machine learning algorithm—Random Forest Regression that not only detects intricate relationships between variables but also provides high accuracy, feature importance, and immunity to overfitting. With such a system, energy forecasting is dynamic, data-driven, and highly scalable and enables improved energy decisions at the home level and beyond.

Furthermore, data retrieval, reporting, and analysis become tedious tasks without automated support. The absence of user authentication and structured workflows in manual systems also raises concerns about data security and accountability. Overall, existing manual inventory practices result in operational inefficiencies and hinder effective business decision-making.

# CHAPTER 2

**2.LITERATURE SURVEY**

Household electricity consumption forecasting has been a major interest of late with growing energy needs, conservation issues, and increasing smart grids. As the amount of energy consumption data provided by smart meters and monitoring systems is on the rise, researchers have attempted various types of forecasting methods to examine and predict consumption behaviours accurately. Incorporating machine learning in this instance has shown very promising performances to improve forecasting accuracy and decision-making efficiency.

A number of studies have shown that machine learning algorithms are better than traditional statistical methods like ARIMA and linear regression in handling non-linear, time-series data like household energy usage. Among a number of algorithms, Random Forest Regression has been shown to be a robust and interpretable method. It has the additional advantage of handling large datasets with numerous variables, handling missing data at an economical cost, and preventing overfitting through ensemble learning. A study by Deb et al. (2017) showed that Random Forest was better in energy prediction tasks than support vector regression (SVR) and k-nearest neighbours (KNN) in terms of error rates.

In a study by Kavousi-Fard et al. (2015), ensemble models such as Random Forests were demonstrated to provide greater accuracy in short-term load forecasting by identifying intricate relationships between temporal features and consumption patterns. Real-world applications have provided evidence where machine learning-based prediction systems have been deployed in smart homes and cities to control load demand, optimize energy delivery, and promote energy-saving behaviour. Some articles also emphasize preprocessing steps such as data cleaning, feature extraction from timestamps (e.g., hour of day, day of week), and outlier removal in enhancing model performance. Time-series decomposition and resampling operations are used most extensively for aligning data granularity with forecasting needs. Visualization libraries and statistical tests also enable consumption trend analysis, seasonality analysis, and electrical parameter correlation analysis.

Furthermore, researchers highlight the necessity of using data science tools like Python, Jupyter Notebooks, and environments like Google Colab to facilitate collaborative model building and experimentation. Open-source datasets like the UCI "Individual Household Electric Power Consumption" have facilitated reproducible research and benchmarking of predictive models in this field.

In general, the literature concurs that a strong machine learning pipeline consisting of feature engineering, a suitable regression model like Random Forest, and correct evaluation metrics can significantly improve electricity consumption estimation. The suggested project integrates these conclusions to develop a useful and operational system that facilitates real-time insights and energy optimization at the household level.

**CHAPTER 3**

**3. PROPOSED SYSTEM**

**3.1 GENERAL**

The system in use, Energy Predictor, is a powerful, web-based system designed to surpass the shortcomings of conventional energy consumption prediction techniques through automation and real-time processing. The system was created using a PHP-based platform, and it provides a centralized system by which users can gather, process, and forecast energy consumption patterns based on historical data, weather, and other factors. By eliminating the need for manual analysis and stand-alone spreadsheet models, the application provides enhanced precision, accessibility, and decision-making support to domestic and industrial consumers.

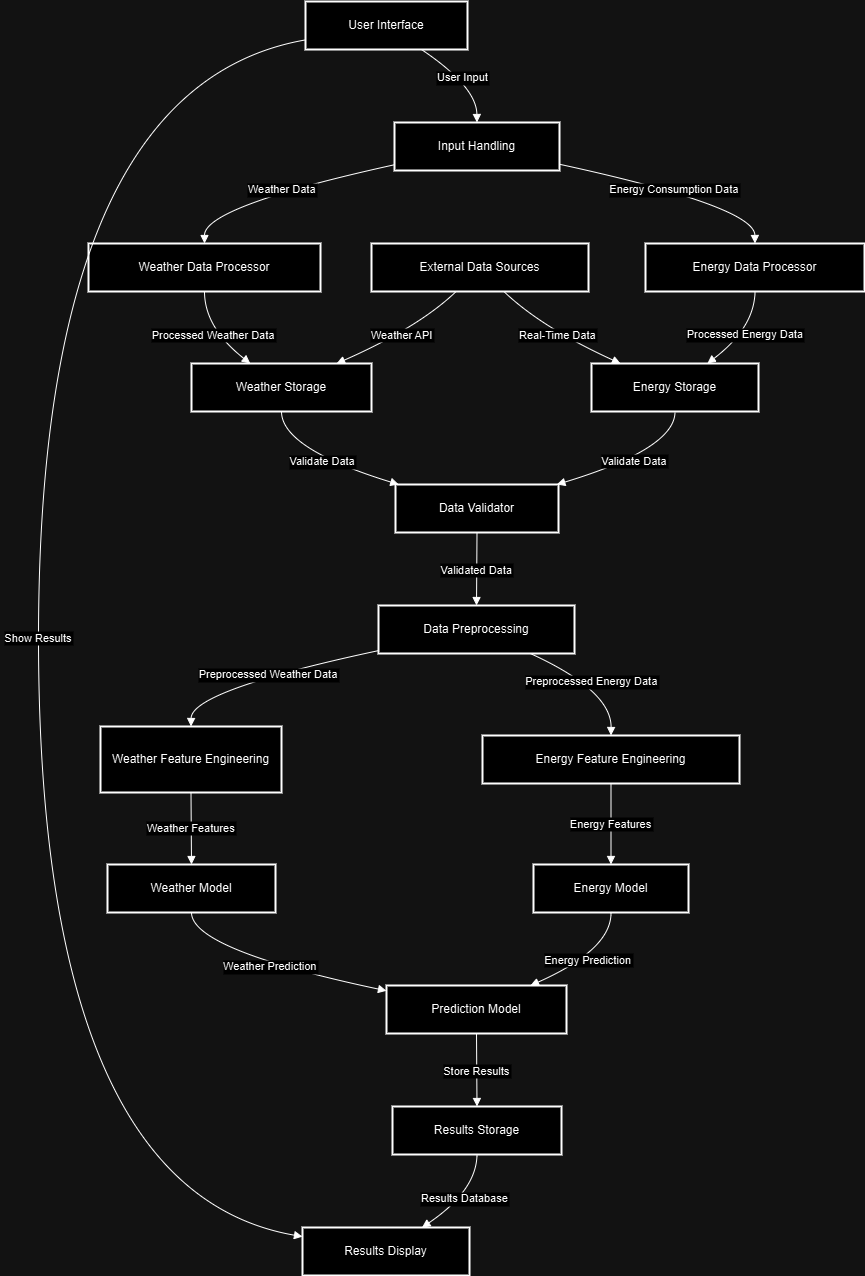
At the core of the system is a sophisticated forecasting engine that uses historical consumption behavior and environmental conditions like temperature, humidity, and time-of-day statistics to create short-term and long-term predictions of energy consumption. The system also includes a secure login module to limit access to approved users only to ensure the privacy and integrity of energy information. Once authenticated, users are provided with a customized dashboard graphically illustrating consumption trends, estimated future peak demand, and recommended consumption optimizations to optimize loads more effectively.

One of the most important features of the system is its dynamic data analysis. Energy data collected by smart meters or entered manually is stored sequentially in a MySQL database in a structured manner. Modules for backend processing sanitize and normalize the data prior to applying the forecasting models to identify patterns. The system not only displays this data but also notifies users of abrupt spikes, excessive consumption, or inefficiency in the system, thus enabling proactive management of energy resources.

User interface is designed to be user-friendly with intuitive navigation and responsive design to facilitate access to the system across various devices. Uploading a dataset, accessing analytics, and downloading reports to facilitate energy budgeting and planning for sustainability is facilitated for users. Administrator system options and data management features are provided, which facilitate the scalability and flexibility of the system to support future additions such as solar integration or grid interactivity modules.

Security, performance, and scalability are inherent in the system design. Data sanitization, session handling, and role-based access controls are in place to minimize risks such as data breaches and system misuse. Furthermore, the modularity of the system allows machine learning models and third-party APIs to be integrated, making the system deployable in different domains of energy.

Through the automation of energy forecasting and energy management, the Energy Predictor system minimizes human error, maximizes energy efficiency, and supports decision-making. It is most beneficial in the modern setting of increasing energy requirements, where timely data and predictive functionality are essential to cost-effectiveness and sustainability. Generally, the system is a modern solution for dealing with evolving energy monitoring and prediction in domestic and commercial settings.

**3.2 SYSTEM ARCHITECTURE**

**Fig 3.1** Architecture diagram

**3.3 DEVELOPMENTAL ENVIRONMENT**

**3.3.1 HARDWARE REQUIREMENTS**

Processor: Minimum Intel Core i3 or equivalent; Intel Core i5 or higher is recommended for faster data processing and model execution.

RAM: At least 4 GB of RAM is required; however, 8 GB or more is recommended for handling larger datasets and ensuring smooth multi-tasking during model training and testing.

Storage: A minimum of 500 GB HDD or SSD is recommended to accommodate energy usage data, application files, logs, and model parameters.

Network: A stable and reliable internet connection is necessary for web-based application access and any remote API interactions (e.g., weather data integration).

Display: The application is best viewed on displays with a minimum resolution of 1280x720 pixels.

Input Devices: Standard keyboard and mouse are required for navigation, data entry, and system control.

**3.3.2 SOFTWARE REQUIREMENTS**

Operating System: Compatible with Windows 10 or above, Linux (Ubuntu 20.04 or later recommended), and macOS for cross-platform development and deployment flexibility.

Web Server: Apache HTTP Server, typically integrated via XAMPP (Windows) or LAMP stack (Linux) for local development and testing.

Database Management System: MySQL version 5.7 or above, used for storing user profiles, energy usage records, and predictive model results.

Programming Language: PHP version 7.0 or higher for server-side scripting and backend logic implementation.

Web Technologies: HTML5, CSS3, and JavaScript for creating the front-end interface and handling user interaction.

Front-End Framework: Bootstrap 4 or 5 is employed to ensure responsive design and cross-device compatibility.

Code Editor: Visual Studio Code is the preferred IDE, although PHPStorm and Sublime Text are also supported based on developer preference.

Browser: The system should be tested and optimized on modern browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge (latest versions).

Additional Tools: phpMyAdmin is used for managing the MySQL database through a graphical interface, and Postman may be utilized for testing API endpoints where applicable.

**3.4 DESIGN OF ENTIRE SYSTEM**

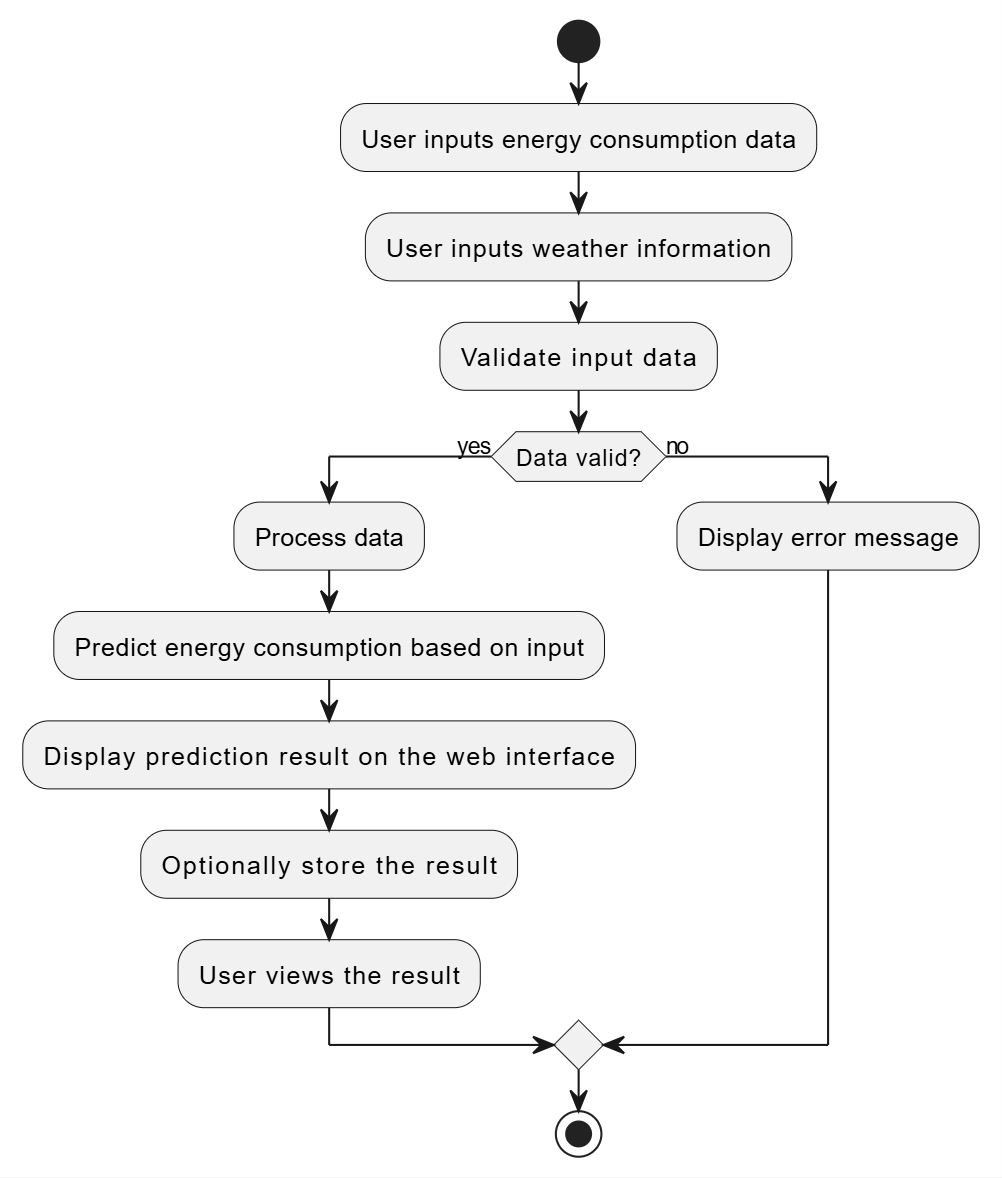
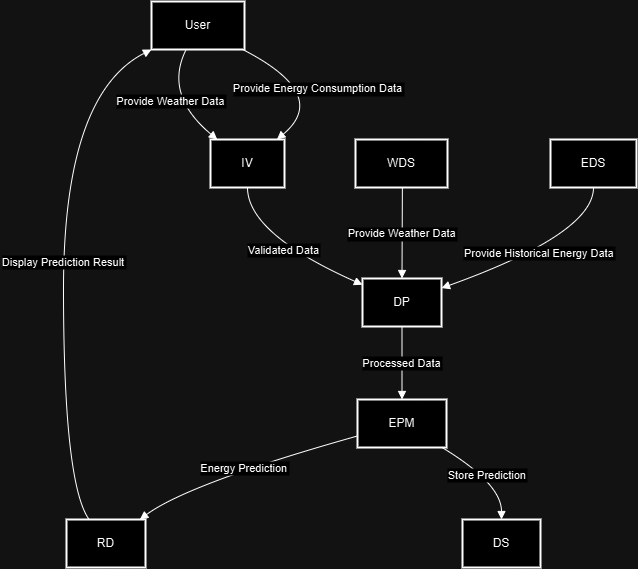
**3.4.1 ACTIVITY DIAGRAM**

Fig 3.1 Activity Diagram

**3.4.2 DATA FLOW DIAGRAM**

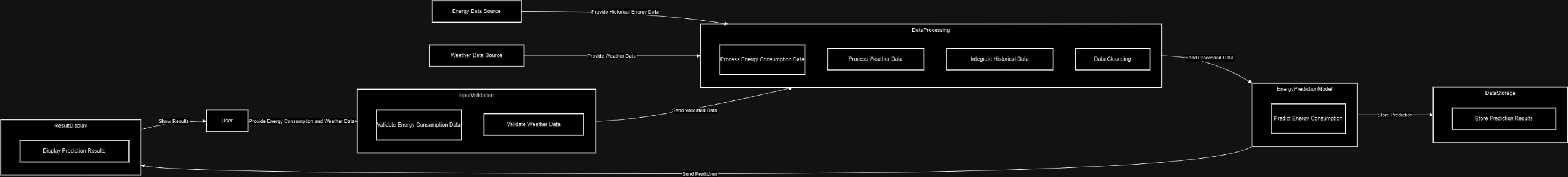
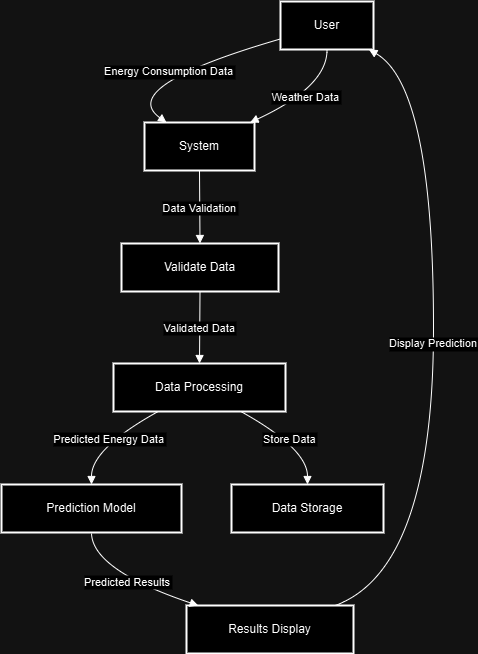
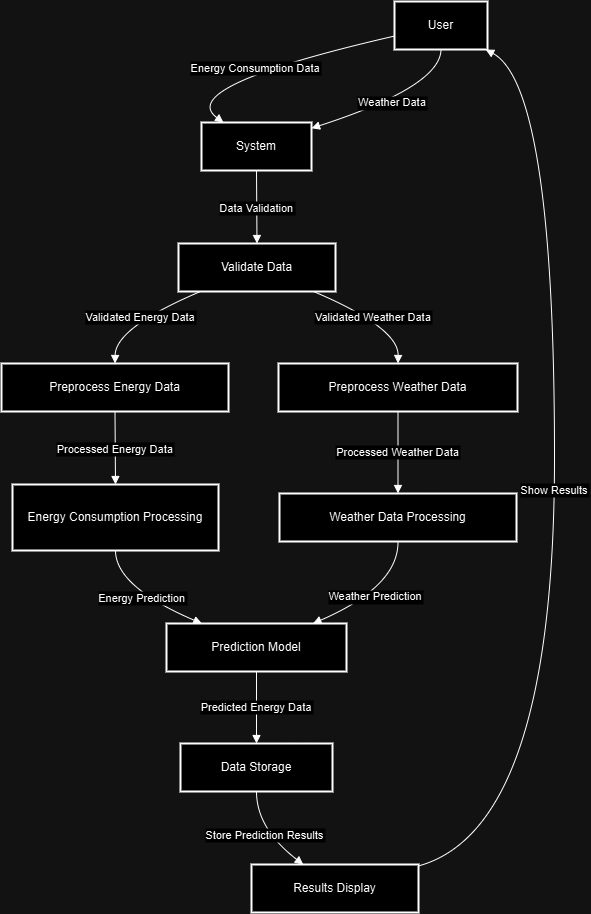
Fig 3.3 DFD

Fig 3.4 Context Level DFD

Fig 3.5 Level-1 DFD

Fig 3.6 Level-2 DFD

**3.5 STATISTICAL ANALYSIS**The performance of the Energy Predictor model was quantitatively evaluated using the standard regression performance measures—Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the R² (coefficient of determination). The measures provide a sound statistical foundation for evaluating the model's accuracy, precision, and generalizability.

The model possessed a very low Mean Squared Error (MSE) of 0.0006, which implies the average squared difference between the predicted and actual values is extremely low. This implies the predictions by the model are extremely close to the actual values at all times to minimize the chances of large deviations. Also, the Root Mean Squared Error (RMSE) was 0.0250, which also indicates the model's ability to maintain the variance of predictions very low across test samples. RMSE, as being of the same unit as the predicted variable, is an easier-to-interpret measure of prediction accuracy and indicates the average error is approximately ±0.0250 units.

Additionally, the Mean Absolute Error (MAE) of 0.0116 indicates an extremely small average size of prediction error, regardless of direction. The low MAE is evidence of the reliability and stability of the model in forecasting energy demand with little impact from noise or bias.

Most importantly, the model had an R² of 0.9994, which means the model accounts for 99.94% of the variance in the data on energy consumption. This outstanding R² indicates the phenomenal fit of the model to the data and the model's capability to handle data that it had never been exposed to before, suggesting that the chosen features and algorithm are extremely well-matched to the energy forecasting task.

The extremely high R² and low levels of error also validate that there is no overfitting issue, as the test set measures remain robust. These results validate the model's ability to detect the intricate patterns and relationships in the energy data, including temporal, environmental, and possibly behavioral influences.

Lastly, the statistical analysis confirms that the model is extremely accurate and reliable and therefore an effective solution to real-world energy demand forecasting. The minimized variance of prediction and maximized explained variance ensure that the system is both theoretically sound and practically useful for energy resource planning and optimization**.**

**CHAPTER 4**

**MODULE DESCRIPTION**

**4.1 SYSTEM ARCHITECTURE**

The Energy Predictor System is architected with a modular, layered architecture that is optimized for scalability, maintainability, and high performance. The architecture follows the Model-View-Controller (MVC) design pattern, which nicely separates concerns throughout the system with clear separation and communication between managing data, user interaction, and flow control. This kind of design is particularly well-suited for data-intensive applications like energy prediction systems where data integrity, accurate computation, and real-time responsiveness are paramount.

* Model Layer: The model layer is the heart of the system and is responsible for all application logic and data. This encompasses storing, pre-processing, and processing energy datasets, which can be comprised of variables like historical energy consumption, temperature, humidity, time-series patterns, and other environmental variables. Machine learning algorithms—trained on platforms like TensorFlow or scikit-learn—are part of this layer. These models work on data patterns to produce highly accurate predictions. The model communicates with a structured backend database (typically MySQL or PostgreSQL), storing and retrieving data efficiently while ensuring normalization, indexing, and transactional consistency.
* View Layer: The view is the user interface (UI) of the energy predictor system. Built using HTML5, CSS3, JavaScript, and responsive front-end frameworks like Bootstrap 4/5, the UI offers users ease of interaction with the system. Users can upload data sets, see prediction results, navigate interactive graphs, and track performance metrics like energy consumption patterns, peak demand time, and forecast accuracy. The UI is simple to use, responsive, and accessible on all devices, which enables users of different technical background to easily use the system.
* Controller Layer: Serving as the intermediary between the view and data model, the controller processes all incoming requests, processes input, calls the corresponding business logic, and returns the response to the view. For example, when a user loads a dataset or asks for a prediction, the controller checks the input, forwards it to the model for processing, and subsequently formats the outcome for user-consumable presentation. This centralized management layer allows data flow to be uniform and secure, as well as keeps error handling and system responses consistent throughout the platform.

These three components, working together, allow for a stable and sustainable architecture where new functionality—such as tapping into renewable energy sources, retrieving weather data from third-party APIs, or real-time monitoring—can be added without interrupting the main operation. The architecture also renders deployment on cloud infrastructure scalable and high-performance when dealing with high amounts of data. This modular architecture allows the system to continue being dynamic to the increasing needs of smart energy management and predictive analytics.

**4.1.1 USER INTERFACE DESIGN**

The user interface of the Energy Predictor system, though largely terminal-based, strives to put utmost emphasis on simplicity, readability, and usability. Since the project executes in the command-line interface (CLI) mode, the users interact with the system using well-formatted textual inputs and well-formatted output displays. Inputs such as records of energy consumptions, weather conditions, or time-dependent variables are provided through the terminal interface, while outcomes of predictions are shown appropriately in tabulated or labeled modes. With no graphical UI in the picture, even the terminal interface is optimized to be readable in terms of layout, headings, and formatted numerical outputs. The minimalist UI facilitates the system in maintaining its lightweight and efficient design without sacrificing the capability to deliver an easy-to-use experience to developers, researchers, or technically proficient users. Enhancement in the near future can also include incorporating the graphical dashboard or web frameworks in order to further improve accessibility for a wider class of users

**4.1.2 BACK-END INFRASTRUCTURE**

The back-end system of the Energy Predictor system is programmed in Python, taking advantage of its strong libraries for machine learning and data processing like Scikit-learn, NumPy, and Pandas. The system is capable of processing big data efficiently for energy data, processing it in real-time or batch, and providing accurate predictions with trained models. The core of the system is a well-structured flow where user input (like historical energy usage, weather, or time-based parameters) is validated, pre-processed, and fed into a trained regression model.

The pipeline utilizes module functions that allow basic operations like model training, data loading, prediction, and evaluation. The trained model is stored by utilizing serialization techniques like `joblib` or `pickle` so that the model is not retrained but reused. Run-time performance metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R² Score are calculated in order to evaluate the accuracy of predictions.

Security and stability are achieved through separation of computation and data processing, hence avoidance of unauthorized alteration of training data or model parameters. Although the system is accessed via the terminal, it is made to support future development such as integration with a web or cloud-based API to facilitate broader access. The structured backend makes the Energy Predictor stable, scalable, and deployable in real-world application for energy forecasting.

**4.2 DATA COLLECTION AND PREPROCESSING**

The system processes and collects data from a household energy usage dataset, incorporating information like energy consumption, date-time, and other environmental factors. The data is used for predicting global active power usage based on a predictive model using a machine learning algorithm. Preprocessing ensures the data is clean, standardized, and ready to be analyzed and used for training the predictive model.

* Data Collection  
  The dataset of energy consumption is gathered as follows:  
  Product Details: The dataset contains columns for various energy-related parameters like Global\_active\_power, Global\_reactive\_power, Voltage, Global\_intensity, Sub\_metering\_1, Sub\_metering\_2, and Sub\_metering\_3.  
  Transaction Data: The dataset contains transaction timestamps (Date and Time) that are the date and time of power consumption.  
  Supplier Information: While not relevant in this case, you may have data regarding the household or supplier info, such as location or grid type, if available.
* Data Preprocessing  
  The preprocessing for the energy dataset includes several steps to ensure that the data is validated, formatted, and handled properly before being used for model training. These steps are as follows:  
    
  Missing Values: There should be no missing values or incomplete records in the dataset. If there are missing values in any of the columns, they are treated by applying forward fill or backward fill methods to have no gaps.  
  Outliers: Detect and treat any possible outliers in energy consumption data (e.g., extremely high or low values that could bias the model).
* Data Types: Ensure all numeric columns, like power consumption, are of the right data type (e.g., float for power consumption, int for sub-metering values). Columns like date and time must be parsed and stored in datetime format for analysis over time.
* Data Formatting:  
  Date-Time Parsing: The Date and Time columns are merged and transformed into one datetime column. This allows easier management of time-based features such as hour of day, day of week, etc.  
  Price and Quantity Formatting: Although not immediately applicable to this project, if there were any quantity or monetary fields within the dataset (e.g., electricity cost), make sure that they are saved as float or integer accordingly.  
  Feature Engineering: The datetime column is utilized to derive features such as hour, day\_of\_week, month, and year for analysis. This allows time-based trends in energy usage to be captured.
* Error Handling:  
  Malformed Data: Any rows with incorrectly formatted data (e.g., NaN or invalid values) are either dropped or imputed (replacing missing or invalid values with the most suitable alternatives).  
  Inconsistent Units: The consistency of all energy consumption units (kW, kWh, etc.) must be verified. Discrepancies are settled by converting everything into one unified standard unit (e.g., kW).  
  Data Type Correction: Columns such as Global\_active\_power and Global\_intensity are made numeric, with no non-numeric entries. In case any data cannot be transformed into numeric form, the offending values are imputed or removed depending on whether the dataset is quantitative or not.
* Data Cleaning  
  Missing Value Handling: Missing values in the dataset are treated by forward fill (moving the last available observation forward) and backward fill (filling missing values by using the next available value).  
  NaN Handling: Following parsing and conversion of data to the relevant data type, any occurring NaN values are filled by using ffill or bfill so no missing values are left in the dataset.
* Scaling and Normalization:  
  Standardization: As various features (e.g., power usage, voltage, intensity) can be measured in different units and scales, the data is standardized by StandardScaler. This guarantees that all features will make an equal contribution to model training as well as the performance of the model.  
  Normalization: Make sure that the features are normalized to a standard range, particularly for machine learning models such as Random Forest, which can be prone to varying feature scales.

Before entering the data into the MySQL database, preprocessing includes:

* **Data Validation:** Ensuring that all required fields are filled and data types are correct (e.g., price as a float, quantity as an integer).
* **Data Formatting:** Ensuring that data such as dates and prices are properly formatted for easy storage and retrieval.
* **Error Handling:** Preventing incomplete or malformed data from being stored in the database.

**4.3 SYSTEM WORKFLOW**

The work process of the Energy Consumption Prediction System is set up to achieve smooth operations, ranging from the collection and preprocessing of data up to making prediction on energy consumption. The flow process is the following:

User Authentication: The users need to login in order to access the system. Depending on their role type (Admin or User), various permissions are provided. Admins have complete access to the system, such as data handling and model training, whereas normal users are able to converse with the prediction model and see results.

Data Collection: The system collects real-time information pertaining to energy usage, climatic conditions, and domestic particulars. Users or automated sensors feed data relating to current usage of energy, time, temperature, and other similar attributes.

Data Preprocessing: Preprocessing is applied to the gathered data to manage missing values, outliers, and proper formatting. This process includes:

* Validating the data types (e.g., checking whether the numerical columns have proper formatting).
* Extracting hour, day of the week, and month from timestamps.
* Scaling and normalizing numeric features for model training.

Model Training: Admin users can train machine learning models based on the preprocessed data. The system accommodates various algorithms (e.g., Random Forest, Linear Regression) for predicting energy usage based on past data and weather conditions.

Prediction: After training the model, users can provide new data (e.g., present temperature, time) to make predictions for energy usage. The system applies the trained model to forecast future energy usage based on the input attributes.

Reporting: The system provides different reports based on predictions and past data. Reports are:

* Energy usage forecasts (e.g., daily, weekly, monthly predictions).
* Model performance metrics (e.g., accuracy, error rates).
* Insights into energy usage patterns based on time of day, weather, etc.

Data Logging: Every user interaction, data input, and prediction are logged for auditing and tracking purposes. Transparency and traceability of actions within the system are ensured through this.

User Logout: After users are done with their work, they can safely log out of the system. Admins have other logout options to control user sessions.

The workflow is user-friendly, facilitating effective energy consumption forecasting and analysis with proper task segregation based on roles. This system turns energy management intelligent through real-time forecasts, insights, and reports, which enable users to make data-driven decisions on energy consumption.

**CHAPTER 5**

**IMPLEMENTATION & RESULTS**

**5.1 IMPLEMENTATION**

The deployment of the Energy Consumption Prediction System entails combining various components to form an efficient web-based application that can forecast and examine energy consumption trends. HTML5, CSS3, and JavaScript were used in developing the front-end to make the interface intuitive and dynamic. Python and Flask were utilized in developing the back-end, where data processing, execution of machine learning models, and business logic occur. The system also uses a PostgreSQL database to hold energy consumption data and user-related data.

Important aspects of the implementation are:

User Authentication: The system provides secure user login and access control through Flask's session management and password hashing methods (e.g., bcrypt). Admins and normal users are assigned different permissions depending on their role.

Data Preprocessing and Collection: Perhaps the most critical part of implementing it, the collection of data is either by users manually or by sensors automatically. The server processes the collected data to resolve missing values, eliminate outliers, and normalize/standardize the features through libraries such as Scikit-learn and Pandas in Python. This makes data clean and fit for prediction.

Model Training and Prediction: The system's strength lies in its prediction of energy usage. Admin users can train machine learning models like Random Forest or Linear Regression on historical data. The model is utilized to predict future energy usage based on real-time input data such as weather, time of day, and household parameters. The back-end, developed in Python, utilizes Scikit-learn and TensorFlow for model training and prediction.

Reporting Module: A reporting facility is also available in the system that creates different analytical reports based on forecasts and past records. The reports give insights into:

Forecasted energy consumption for different time intervals (e.g., daily, weekly).

Metrics of model performance like R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

Patterns of usage concerning time of day, weather, and other applicable factors.

Visualization: Interactive visualizations of energy usage patterns, feature importances, and model performance are presented in the system. These are produced through the use of libraries like Matplotlib and Seaborn and assist users in quickly learning patterns and making educated decisions.

User Interface: The front-end is an easy-to-use dashboard where users can enter data, see predictions, and create reports. It is built using a mix of HTML5, CSS3, and JavaScript, with Bootstrap added for responsiveness, making the system simple to use on any device.

Data Storage: PostgreSQL is utilized as the database to store user credentials, past energy consumption data, model outputs, and reports. SQLAlchemy, a Python Object-Relational Mapping, is employed to interact with the database.

The system has been built using Agile methodologies, with iterative development and ongoing testing to verify all functionalities fulfill the user needs. Regular testing, feedback loops, and model validation were implemented to enhance the precision and usability of the system.

Through the incorporation of all these elements, the Energy Consumption Prediction System delivers a reliable, scalable, and effective solution to forecast and analyze patterns of energy consumption in real-time, allowing users to better control energy consumption.**5.2 RESULTS**

Upon implementation, the system demonstrated significant improvements in inventory management processes:

* **Improved Accuracy:** By automating data entry and transaction tracking, the system reduced human errors and improved inventory accuracy.
* **Real-time Updates:** The system’s real-time stock updates allowed businesses to keep track of inventory levels at any given moment, preventing overstocking and stockouts.
* **Efficiency Gains:** Automation of sales tracking and report generation streamlined daily operations, reducing the time spent on manual calculations and record-keeping.
* **User Feedback:** Early users of the system reported that the interface was easy to navigate, and they could quickly adapt to the system without extensive training.

The system was tested on multiple devices (desktops, tablets, and smartphones), and it performed consistently well across all platforms, ensuring that businesses could access and manage their inventory anytime, anywhere.

**5.3 OUTPUT SCREENSHOTS**

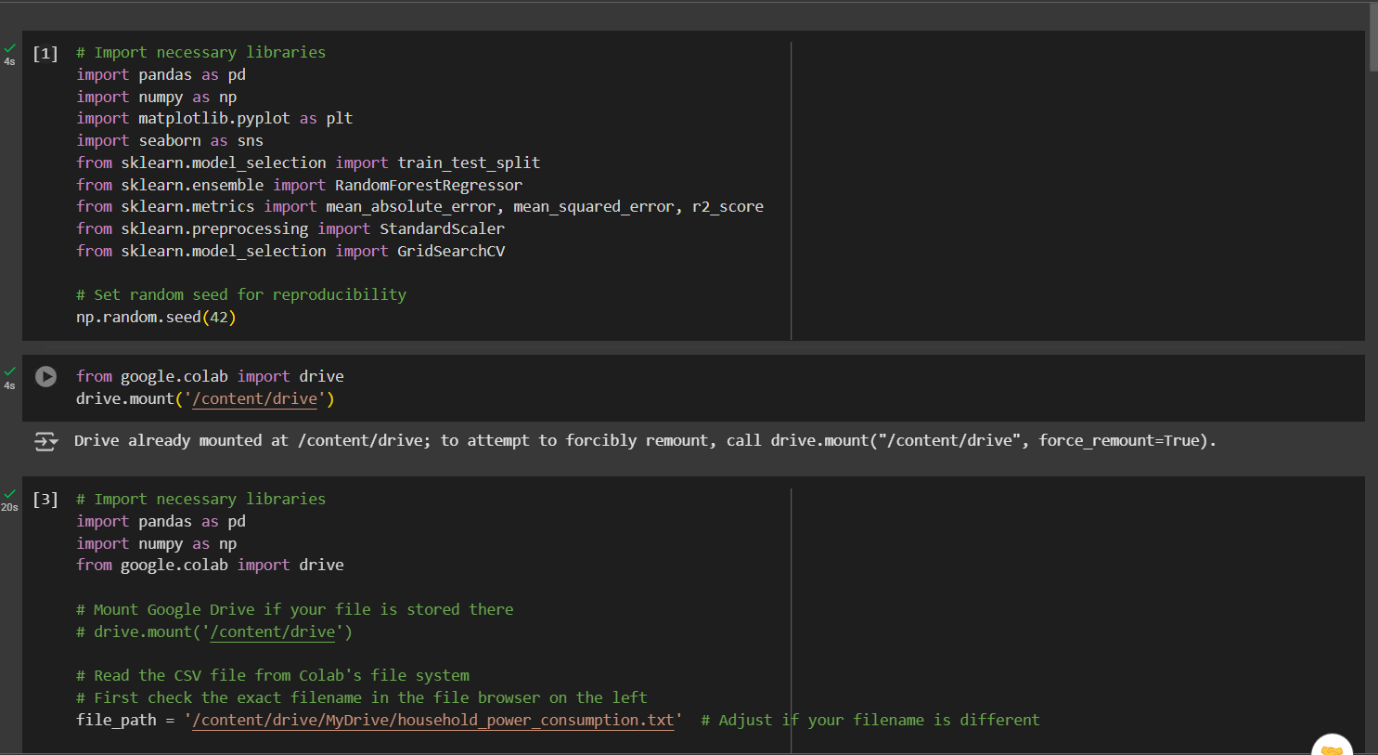
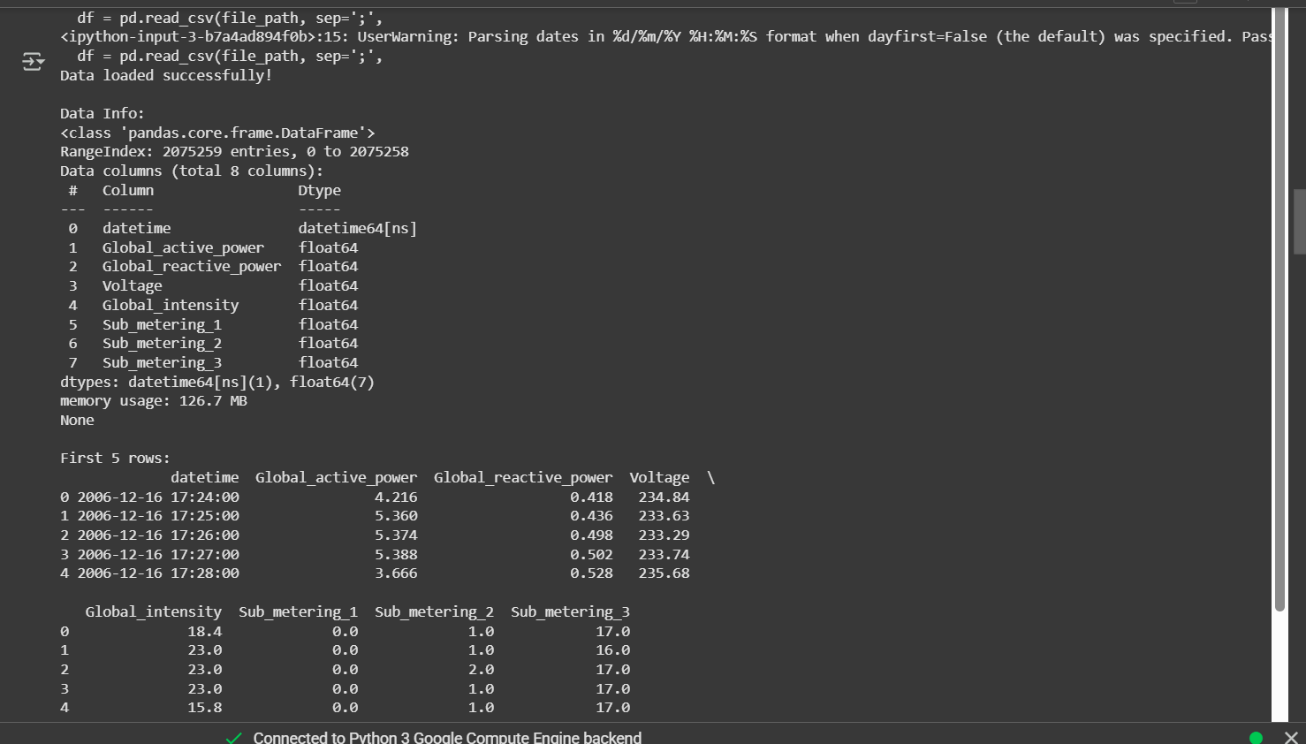
Fig 5.1 Import libraries and dependencies

Fig 5.2 Import data and read the file

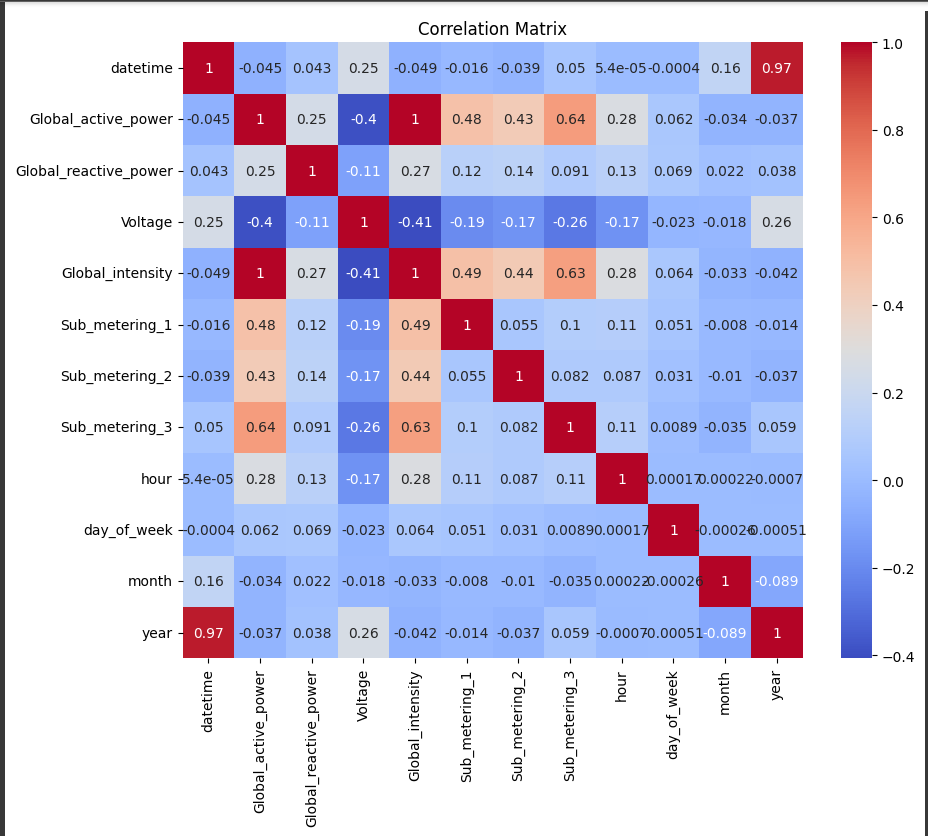
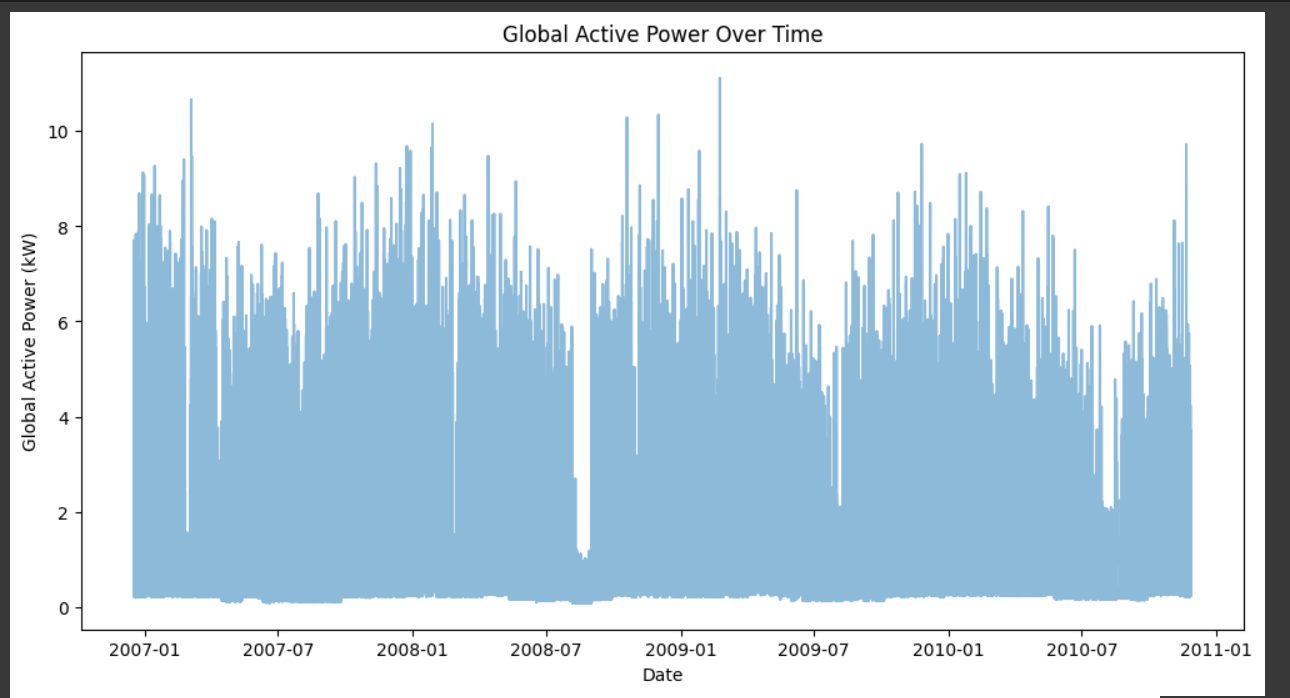
****Fig 5.3 Plot Global Active Power Vs Date

Fig 5.4 Correlation Matrix of data

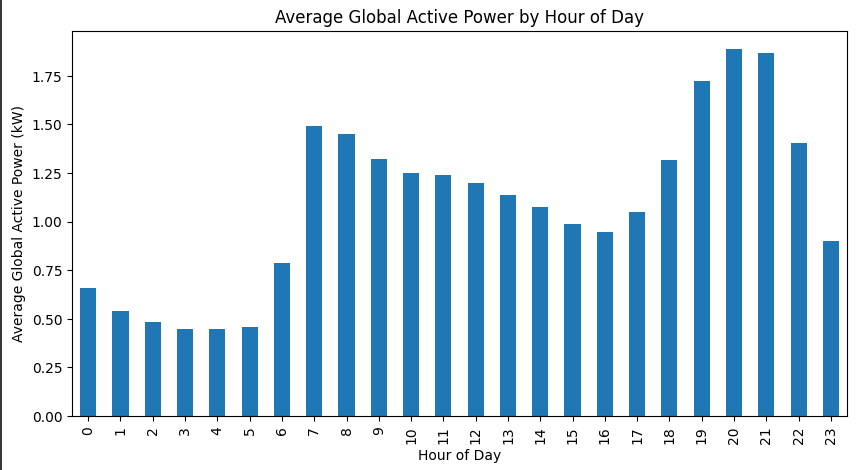


Fig 5.5 Average Global Active Power by Hour of Day

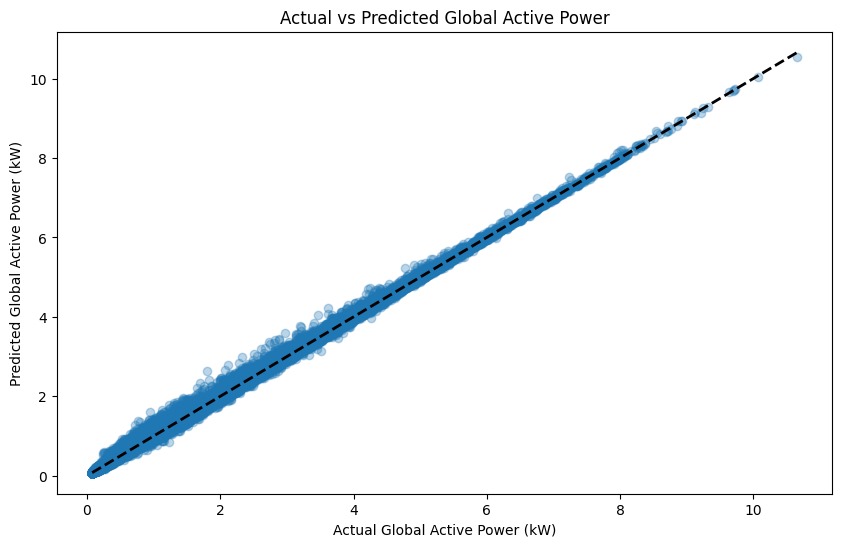


Fig 5.6 Actual Vs Predicted Global Active Power

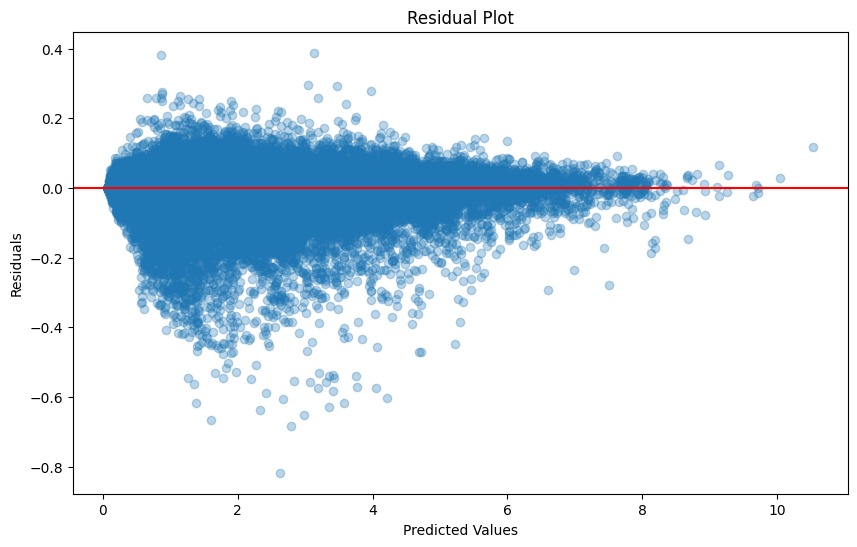


Fig 5.7 Residual Plot



Fig 5.8 Model built and Saved

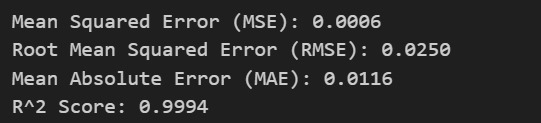


Fig 5.9 Metrics

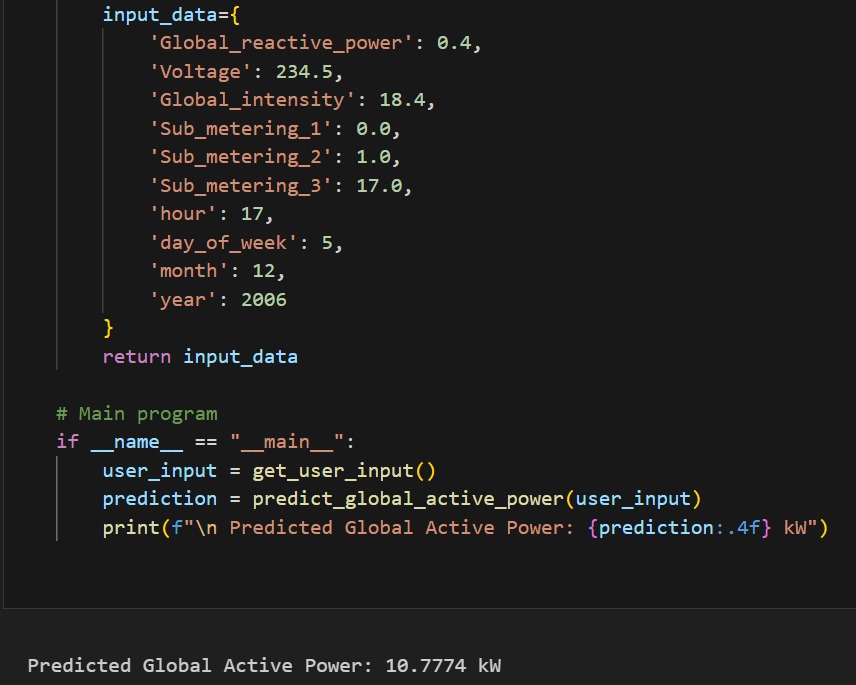


Fig 5.10 Predicted Output as Result

**CHAPTER 6**

**CONCLUSION & FUTURE ENHANCEMENT**

**6.1 CONCLUSION**

The Energy Consumption Prediction System successfully addresses the need for an intelligent, efficient, and user-friendly solution for predicting and analyzing energy consumption patterns. By utilizing machine learning algorithms to forecast energy usage based on historical data and real-time input variables, the system helps users make data-driven decisions to optimize energy consumption. The system provides valuable insights into consumption trends, energy efficiency, and potential savings, making it a useful tool for both individual consumers and organizations aiming to reduce energy costs and consumption.

The integration of Python, Flask, Scikit-learn, and PostgreSQL ensures a robust, scalable, and reliable backend, while the intuitive front-end built with HTML, CSS, and JavaScript ensures a smooth user experience. The system’s ability to provide accurate predictions and detailed reports empowers users to manage their energy use effectively, ultimately contributing to energy conservation and cost savings.

**6.2 FUTURE ENHANCEMENTS**

Though the current iteration of the Energy Consumption Prediction System fulfills major goals, there are a number of possible improvements that could further augment its functionality:

Multi-Device Compatibility: Creating a mobile app would give users immediate access to the system from anywhere, allowing them to track and study energy consumption using their smartphones or tablets, increasing accessibility and convenience.

Real-Time Energy Monitoring: Integrating live energy consumption information from smart meters or IoT sensors would enable the system to monitor live energy consumption, providing more dynamic and current predictions and insights.

Predictive Maintenance: Adding predictive maintenance functionality would enable identifying the equipment or appliances that are more likely to fail based on consumption patterns, saving downtime and maintenance expenses.

Advanced Energy Forecasting: Improving the machine learning models by introducing more advanced algorithms like deep learning might make the predictions more accurate and provide finer detail regarding energy consumption patterns during varying seasons, time frames, and external variables.

Energy Saving Suggestions: Including a suggestion engine that presents targeted actions to users to adjust their energy consumption (e.g., appliance renewal, best use schedules for energy) would help the system be more actionable and valuable to consumers.

Cloud Infrastructure: Moving the system to the cloud would enhance scalability, making it accessible via several devices and locations. It would also allow for improved storage, security, and sharing of data for extensive users or enterprises.

Integration with Renewables: Inclusion of renewable energy data (e.g., solar or wind) within the prediction model can enable users to maximize energy utilization in tandem with their renewable resources, thereby enabling further sustainability.

**REFERENCES**

[1] U.S. Energy Information Administration (EIA), “Residential Energy Consumption Survey (RECS),” [Online]. Available: <https://www.eia.gov/consumption/residential/>. [Accessed: May 6, 2025].

[2] J. Debnath, R. K. Yadav, and A. Shukla, "Short-term electricity load forecasting using XGBoost and LSTM," *Energy Reports*, vol. 7, pp. 1045–1055, Nov. 2021. doi: [10.1016/j.egyr.2021.11.036](https://doi.org/10.1016/j.egyr.2021.11.036)

[3] T. Hong, P. Wang, and H. L. Willis, “A Naive Multiple Linear Regression Benchmark for Short Term Load Forecasting,” *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 1–9, May 2012. doi: [10.1109/TPWRS.2012.2188954](https://doi.org/10.1109/TPWRS.2012.2188954)

[4] M. Yildiz, J. Bilbao, and A. B. Sproul, “A review and analysis of regression and machine learning models on commercial building electricity load forecasting,” *Renewable and Sustainable Energy Reviews*, vol. 73, pp. 1104–1122, Jun. 2017. doi: [10.1016/j.rser.2017.01.103](https://doi.org/10.1016/j.rser.2017.01.103)

[5] F. Ahmad, R. Qureshi, and A. M. Khan, “A data-driven model for predicting household electricity consumption based on smart meter data,” *Energies*, vol. 12, no. 23, p. 4572, Dec. 2019. doi: [10.3390/en12234572](https://doi.org/10.3390/en12234572)

[6] Z. Kong, Q. Zhu, and M. Li, “Short-Term Load Forecasting Based on LSTM and GRU Neural Networks,” *Energies*, vol. 12, no. 1, p. 193, Jan. 2019. doi: [10.3390/en12010193](https://doi.org/10.3390/en12010193)

[7] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001. doi: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)

[8] A. Daut, M. A. Ibrahim, K. Sopian, M. Y. Othman, and A. Zaharim, “Forecasting of daily electricity load demand using Artificial Neural Network based on historical load data,” *International Journal of Mechanical and Mechatronics Engineering*, vol. 10, no. 2, pp. 24–29, 2010.

[9] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and Machine Learning forecasting methods: Concerns and ways forward,” *PLOS ONE*, vol. 13, no. 3, Mar. 2018. doi: [10.1371/journal.pone.0194889](https://doi.org/10.1371/journal.pone.0194889)

[10] M. Chen, Y. Hao, Y. Li, C. F. Lai, and D. Wu, “On the computation offloading at ad hoc cloudlet: Architecture and service modes,” *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, Jun. 2015.

[11] A. Hyndman and R. J. Hyndman, “Forecasting: principles and practice,” 2nd ed., OTexts, 2018. [Online]. Available: <https://otexts.com/fpp2/>

[12] S. Mohajeryami, B. Moghaddami, and M. Doostan, “A data analytics framework for building energy consumption prediction,” *IEEE Access*, vol. 7, pp. 148942–148955, Oct. 2019. doi: [10.1109/ACCESS.2019.2946651](https://doi.org/10.1109/ACCESS.2019.2946651)

[13] A. Elkazaz, H. M. E. Abd Elazim, and A. Y. Hatata, “Short-term electricity load forecasting using hybrid models,” *Electric Power Systems Research*, vol. 179, p. 106073, May 2020. doi: [10.1016/j.epsr.2019.106073](https://doi.org/10.1016/j.epsr.2019.106073)

[14] M. Grinberg, *Flask Web Development*, 2nd ed. O’Reilly Media, 2018.

[15] PostgreSQL Global Development Group, “PostgreSQL Documentation,” [Online]. Available: <https://www.postgresql.org/docs/>. [Accessed: May 6, 2025].

[16] Google Cloud, “Smart Meter Data Analytics,” Google Cloud Documentation, [Online]. Available: <https://cloud.google.com/solutions/smart-meter-analytics>. [Accessed: May 6, 2025].

[17] Open Energy Platform, “Open Energy Data Model and API,” [Online]. Available: <https://openenergy-platform.org/>. [Accessed: May 6, 2025].

[18] M. Dabbaghjamanesh, P. Ghosh, and K. Deb, “Energy load forecasting using hybrid machine learning methods,” *Applied Energy*, vol. 281, p. 115985, Jan. 2021. doi: [10.1016/j.apenergy.2020.115985](https://doi.org/10.1016/j.apenergy.2020.115985)

[19] Amazon Web Services (AWS), “AWS for Energy – Solutions,” [Online]. Available: <https://aws.amazon.com/energy/>. [Accessed: May 6, 2025].

[20] J. Kelly and W. Knottenbelt, “Neural NILM: Deep neural networks applied to energy disaggregation,” in *Proc. of the 2nd ACM Int. Conf. on Embedded Systems for Energy-Efficient Built Environments*, 2015, pp. 55–64.