FUDMA Journal of Engineering and Technology
ISSN: 3092-9385
https://sites.google.com/fudutsinma.edu.ng/fjet
Volume 1, Issue 2, 2025, 342-353

# Development of an Edge-Enabled IoT Smart Energy Meter with Artificial Intelligence (AI)-Based Load Prediction for Device-Level Monitoring

Adekunle O. ADEWOLE[1*], Ayodeji O. ARIYO[2]

[1*]Department of Electrical/Electronics Engineering, Abraham Adesanya Polytechnic, Ijebu-Igbo, Ogun State, Nigeria
[2]Department of Computer Engineering, Abraham Adesanya Polytechnic, Ijebu-Igbo, Ogun State, Nigeria

[1*]kunleadewole2008@gmail.com, [2]ayodejiariyo2013@yahoo.com

***Abstract***

*The growing demand for intelligent energy management has accelerated the integration of the Internet of Things (IoT), edge computing, and Artificial Intelligence (AI) in smart metering. This paper presents the development of an edge-enabled IoT smart energy meter with AI-based load prediction for device-level monitoring. The system employs a PZEM-004T sensor for measurement of voltage, current, power, energy, and frequency, while a Raspberry Pi serves as the edge device for local processing and storage. A machine learning framework was trained on three months of data and evaluated using k-fold cross-validation. Results show that Linear Regression achieved the highest accuracy (R²: 0.993±0.001, MAE: 0.041, RMSE: 0.051) with minimal training (0.0017s), inference time, and model size (0.05 MB). Random Forest also performed well (R²: 0.990) but required higher computation, while KNN (R²: 0.920) and LSTM (R²: 0.602) were less efficient. SHAP-based analysis confirmed that temporal and electrical features were the most influential. The best-performing model was deployed on the Raspberry Pi and integrated with a Django-based dashboard for real-time monitoring and predictive analytics, providing a practical and efficient solution for energy management.*

***Keywords:*** *Smart energy meter, edge computing, artificial intelligence, load prediction, energy management.*

## 1.0 Introduction

The global energy landscape is undergoing a profound transformation, driven by rising energy demands, the urgent need for sustainable practices, and rapid advancements in digital technologies [1–3]. Within this evolving context, intelligent energy management systems [4] have emerged as a vital component for optimizing energy consumption, enhancing grid stability, and enabling the large-scale integration of renewable energy sources. The convergence of the Internet of Things (IoT), edge computing, and Artificial Intelligence (AI) is central to this transformation, offering unprecedented opportunities to develop more efficient, resilient, and responsive energy infrastructures [5].

IoT, with its pervasive network of interconnected devices, facilitates real-time data acquisition across multiple points in an energy system. Such granular data provides critical insights into consumption patterns, operational efficiencies, and opportunities for optimization [6]. However, the vast volume and velocity of IoT-generated data challenge traditional centralized cloud-based processing models, giving rise to issues of latency, bandwidth limitations, and data privacy concerns [7]. Edge computing mitigates these challenges by relocating computational capabilities closer to the data source, i.e., at the "edge" of the network. Processing data locally reduces dependence on cloud infrastructure, thereby minimizing latency, conserving bandwidth, and enhancing data privacy. This capability is particularly valuable in applications requiring real-time decision-making, such as smart energy management [8–10].

Artificial Intelligence, particularly machine learning, provides the analytical capacity to extract actionable insights from large-scale IoT datasets. AI algorithms can identify hidden patterns, forecast future trends, and optimize system performance. In energy management, AI-based load prediction is especially critical for forecasting demand, enabling proactive resource allocation, and supporting demand-side management strategies [11].

Despite the increasing adoption of smart meters, their current functionality remains limited in realizing the full potential of intelligent energy management. Conventional smart meters primarily function as data acquisition and communication devices, transmitting raw consumption data to centralized systems for subsequent analysis [12,13]. This approach presents several limitations. First, most smart meters lack on-device intelligence due to insufficient computational power and memory, delaying real-time insights since data must first be processed in the cloud [14]. Second, continuous transmission of fine-grained consumption data raises significant privacy concerns, as such data may reveal sensitive information about occupants' daily routines [15,16]. Third, the centralized architecture places heavy burdens on communication networks; as the number of meters increases,

Received: 07/08/2025; Revised: 31/08/2025; Accepted: 01/09/2025; Published: 15/09/2025

data volumes can overwhelm infrastructure, cause congestion, and escalate operational costs, thereby limiting scalability in future smart grids projected to integrate billions of devices [17,18]. Finally, traditional meters remain inadequate for demand-side management because, without embedded intelligence or predictive capability, they cannot autonomously adapt to dynamic factors such as real-time pricing, grid demand, or renewable energy availability, thereby constraining their utility for both consumers and service providers [19,20]. These challenges highlight the urgent need for a new generation of smart energy meters with embedded intelligence and edge-computing capabilities. By shifting computation closer to the data source, such devices can reduce latency, safeguard privacy, alleviate network congestion, and enable autonomous, adaptive energy management [21].

This paper addresses these challenges by presenting the development of an edge-enabled IoT smart energy meter with AI-based load prediction for device-level monitoring. The proposed system integrates a PZEM004T sensor for precise electrical parameter measurement and a Raspberry Pi as a computationally capable edge device. This architecture supports real-time data acquisition, local processing, and storage, thereby reducing reliance on centralized cloud infrastructure. A machine learning framework was deployed on the Raspberry Pi to enable short-term load forecasting. Multiple models—including Linear Regression, K-Nearest Neighbors (KNN), Random Forest, and Long Short-Term Memory (LSTM)—were rigorously evaluated using three months of locally collected energy data. Features such as hour, day of the week, and historical consumption were considered for prediction. To enhance transparency and interpretability, SHAP analysis was employed to determine the influence of temporal and electrical parameters on model outputs. A comparative evaluation was conducted based on Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and the coefficient of determination ($R^2$). In addition, a Django-based web dashboard was developed to provide real-time monitoring and intuitive visualization of both consumption patterns and predictive insights. This interface enhances user engagement and supports informed decision-making for energy management. Overall, the developed system offers a cost-effective solution for residential and institutional energy monitoring, while strengthening demand-side management practices. It thus contributes to the advancement of more intelligent, sustainable, and resilient energy ecosystems.

## 2.0 Related Work

Smart energy meters represent a significant evolution from traditional electromechanical meters, offering advanced functionalities such as real-time energy consumption monitoring, bidirectional communication, and remote control capabilities. The primary objective of smart metering infrastructure is to improve energy efficiency, enhance grid reliability, and empower consumers with greater control over their energy usage. Early developments in this field focused primarily on automated meter reading (AMR) and advanced metering infrastructure (AMI), which enabled utilities to remotely collect consumption data with greater efficiency [22,23].

Over time, smart meters have evolved beyond simple data collection. Modern devices increasingly incorporate communication technologies such as Wi-Fi, ZigBee, and cellular networks, enabling seamless interaction between meters, utilities, and consumers. This connectivity supports advanced applications including dynamic pricing, demand response, and fault detection [24]. However, many existing solutions remain constrained by their reliance on centralized, cloud-based data processing for advanced analytics. Such dependence introduces challenges such as high latency, increased network traffic, and potential single points of failure, particularly in large-scale deployments [23,25]. To overcome these limitations, there is a growing demand for intelligent and autonomous meters capable of performing local data processing and decision-making.

Edge computing has emerged as a transformative paradigm within the Internet of Things (IoT) [26], addressing limitations of cloud-centric architectures by shifting computational resources closer to data sources. In this model, data processing, analysis, and storage occur locally at or near the point of generation, rather than being transmitted to distant cloud servers. This approach offers several advantages for smart energy management. First, latency is significantly reduced, enabling near real-time decision-making in critical applications such as demand-side management and grid stabilization [27,28]. Second, bandwidth utilization is optimized, as raw data can be pre-processed, filtered, and aggregated at the edge before transmission, thereby reducing network congestion and operational costs [29]. Third, privacy and security are enhanced since sensitive data can be retained and processed locally, reducing exposure risks associated with cloud transmission [30]. Despite these benefits, edge computing faces challenges including limited computational resources, the complexity of managing distributed systems, and the need for robust multi-layered security. Nevertheless, ongoing hardware and software advances continue to expand the viability of edge devices for applications such as AI-driven load prediction in smart energy meters.

Artificial Intelligence (AI) and machine learning (ML) have transformed load forecasting by providing more accurate and robust predictions compared to traditional statistical techniques. Accurate forecasting is critical for efficient energy management, grid operation, and demand-side response strategies [31]. Various ML models have been widely studied for short, medium, and long-term forecasting. Linear Regression, while simple and interpretable, often underperforms in complex non-linear energy consumption scenarios due to its inherent assumption of linear relationships [32]. K-Nearest Neighbors (KNN), an instance-based learning algorithm, can capture non-linearities but suffers from high computational cost in large datasets and sensitivity to distance metrics [33]. Random Forest, an ensemble decision-tree method, has demonstrated strong predictive performance, robustness against overfitting, and the ability to handle high-dimensional datasets, making it a strong candidate for load prediction [34]. Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, excel at modeling sequential dependencies, making them highly effective for time-series energy consumption forecasting [35]. Recent studies also highlight the effectiveness of deep learning and hybrid models for example CNN-LSTM, in capturing both temporal and spatial dependencies in energy data [36]. However, deploying such sophisticated models on resource-constrained edge devices remains a considerable challenge, necessitating model optimization and lightweight inference approaches.

Although substantial progress has been made independently in smart metering, edge computing, and AI-based load prediction, a notable research gap remains in the holistic integration of these technologies into a single, edge-enabled smart energy meter capable of real-time monitoring and predictive analytics. Existing smart meters typically lack sufficient on-device intelligence and continue to rely heavily on cloud platforms, raising issues of latency, scalability, and privacy. While edge computing offers a potential solution, its combined application with advanced AI models on resource-constrained devices such as Raspberry Pi is still under active investigation. Prior studies have successfully demonstrated AI-based forecasting, but these implementations often rely on powerful cloud servers or centralized platforms rather than edge deployments. The challenge lies in optimizing ML models for efficient execution within the constraints of edge hardware while maintaining predictive accuracy and reliability.

This research addresses the identified gap by developing an integrated edge-enabled IoT smart energy meter system that incorporates a robust sensor module, a Raspberry Pi as the edge device, and optimized AI models for load prediction. By evaluating and comparing the performance of Linear Regression, KNN, Random Forest, and LSTM models using real-world, locally collected data, this work demonstrates the feasibility of deploying advanced predictive analytics directly at the edge. Furthermore, the system provides a practical framework for enabling autonomous energy monitoring and demand-side management, contributing to the advancement of intelligent, secure, and scalable energy infrastructures.

## 3.0 System Design and Methodology

This section described the architectural design and methodological framework adopted for the development of the edge-enabled IoT smart energy meter with AI-based load prediction. The system was structured to deliver real-time energy monitoring and predictive analytics by exploiting edge computing capabilities, thereby overcoming the inherent limitations of conventional cloud-dependent smart metering systems.

### 3.1 System Architecture

The proposed system architecture was organized into three functional layers: the Sensing Layer, the Edge Computing Layer, and the Application Layer. This modular design enables efficient data acquisition, localized processing, and intuitive user interaction, ensuring seamless operation across the entire system. A schematic representation of the architecture is presented in Figure 1.

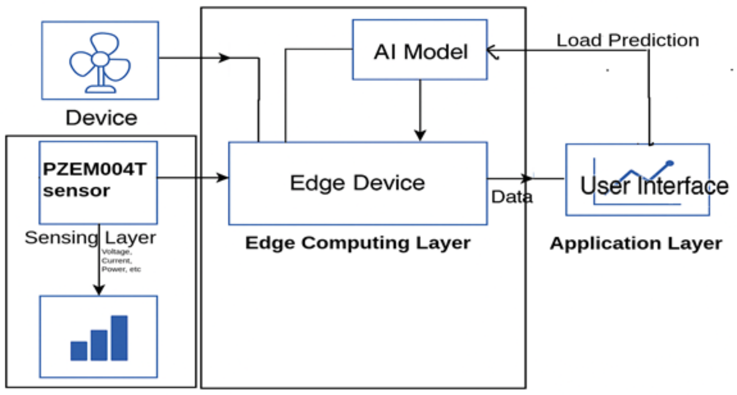


Figure 1: System Architecture of Edge-Enabled IoT Smart Energy Meter

**I. Sensing Layer:** This layer is responsible for the accurate measurement of electrical parameters. It primarily consists of the PZEM004T sensor, which interfaces directly with the electrical circuit to collect raw data such as voltage, current, power, energy, and frequency.

**II. Edge Computing Layer:** The core of our intelligent energy meter lies in this layer, facilitated by a Raspberry Pi 5. The Raspberry Pi acts as the edge device, performing several critical functions: data acquisition from the PZEM004T sensor, local data processing and storage, and running the AI-based load prediction model. By processing data at the edge, this layer significantly remove latency and bandwidth requirements, while enhancing data privacy.

**III.Application Layer:** This layer provides the interface for users to interact with the system. It is implemented as a Django-based web dashboard, accessible via a web browser. The dashboard displays real-time energy consumption data, visualizes historical trends, and presents the load prediction forecasts generated by the AI model. It also allows for potential configuration and management of the system.

This architecture ensures that critical data processing and predictive analytics occur close to the source, enabling rapid response and eliminating the burden on cloud infrastructure, while still providing a comprehensive overview to the end-user.

### 3.2 Hardware Components

The selection of hardware components was driven by the need for accuracy, reliability, and the ability to perform edge computing tasks efficiently. Hardware components used in the design are Rasberry Pi 5, RS485-TTL module, PZEM-004T, 5V power supply Module and Printed Circuit Board. The circuit diagram of the project is shown in figure 2.

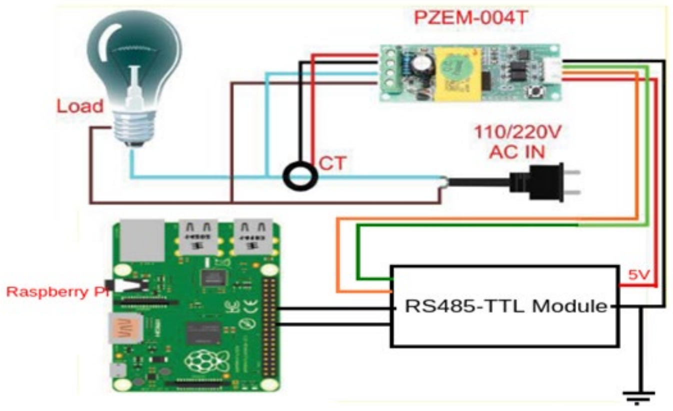


Figure 2: Circuit Diagram of Edge-Enabled IoT Smart Energy Meter

#### 3.2.1 PZEM004T Sensor

The PZEM004T is a highly versatile and cost-effective AC multi-function electric energy power monitor module. It is specifically chosen for its capability to accurately measure a comprehensive set of electrical parameters crucial for energy monitoring [23]. The sensor can measure AC voltage (80-260V), current (up to 100A with a current transformer - CT), active power, energy consumption (kWh), and frequency (45-65Hz). This comprehensive data collection is essential for detailed energy analysis and load profiling. Also,the PZEM004T can communicates via a TTL serial interface (UART), making it straightforward to integrate with microcontrollers or single-board computers like the Raspberry Pi. This serial communication allows for reliable and efficient data transfer from the sensor to the edge device. This sensor also provides relatively high accuracy for its price point, making it suitable for residential and small-scale institutional energy monitoring applications. Its robust design ensures reliable operation in typical electrical environments.

#### 3.2.2 Raspberry Pi 5

The Raspberry Pi 5 is a series of small single-board computers that serves as the central processing unit for the Edge Computing Layer. Its compact size, low power consumption, and significant processing power make it an ideal choice for an edge device in this application. The Raspberry Pi is capable of running a full-fledged Linux operating system, providing a powerful platform for executing Python scripts for data acquisition, preprocessing, and running machine learning inference. This enables on-device computation, reducing the need to send all raw data to the cloud. With expandable storage options (microSD card), the Raspberry Pi can store several months of energy consumption data locally. This local storage is crucial for training and retraining the AI model with historical data, as well as for providing data redundancy. The Raspberry Pi offers various connectivity options, including Wi-Fi and Ethernet, allowing it to connect to the local network for dashboard access and, if necessary, to the internet for cloud synchronization or updates. Its GPIO pins facilitate easy interfacing with the PZEM004T sensor through the RS485-TTL module. Lastly, the affordability of the Raspberry Pi makes the overall smart energy meter system a cost-effective solution, particularly for widespread deployment in residential or institutional settings.

## 3.3 Software Components

The software architecture is designed to facilitate seamless data flow from sensing to prediction and visualization.

### 3.3.1 Data Acquisition Module

This module, developed in Python and running on the Raspberry Pi, is responsible for interfacing with the PZEM004T sensor and collecting real-time electrical parameter data. It continuously reads data from the sensor via the serial port, parses the incoming data packets, and stores the extracted voltage, current, power, energy, and frequency values. The module is designed to handle potential communication errors and ensure data integrity. Collected data is timestamped and stored locally on the Raspberry Pi, forming the dataset for the load prediction model.

### 3.3.2 Machine Learning Framework

The machine learning framework encompassed the development, training, evaluation, and deployment of the AI-based load prediction model, and it was implemented in Python using established libraries such as Scikit-learn and TensorFlow/Keras, depending on the model architecture (e.g., LSTM). The framework supported the training of multiple machine learning models, including Linear Regression, K-Nearest Neighbors (KNN), Random Forest, and Long Short-Term Memory (LSTM), using locally collected historical energy consumption data. Feature engineering was applied to derive relevant predictors such as hour of the day, day of the week, and lagged energy consumption values, thereby enhancing the learning process. Following training, each model was rigorously evaluated with regression metrics—Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and the coefficient of determination ($R^2$)—to ensure robust performance assessment and facilitate the selection of the most effective model. The best-performing model was then deployed on the Raspberry Pi by saving the trained model and embedding it within the data acquisition pipeline, enabling real-time load prediction from incoming sensor data and providing actionable insights at the edge.

### 3.3.3 Django-based Web Dashboard

A web-based dashboard was developed using the Django framework to provide a user-friendly interface for monitoring and visualizing energy data and predictions. Django was chosen for its robustness, security features, and rapid development capabilities. Figure 3 shows the dashboard web interface with different analytics display.

The dashboard shows real-time readings from the PZEM004T sensor, providing an immediate overview of current energy consumption. This is achieved by fetching data from the local database on the Raspberry Pi. Users can be able to view historical energy consumption trends over various periods (e.g., daily, weekly, monthly) through interactive charts and graphs, which helped in identifying consumption patterns and anomalies. The dashboard visualizes the load predictions generated by the AI model, allowing users to anticipate future energy demands. This feature is crucial for effective demand-side management and energy planning. The dashboard was designed to be intuitive and responsive, ensuring accessibility from various devices (desktops, tablets, smartphones) within the local network. Figure 3 shows the Django-based web dashboard as deployed on rasberry pi.

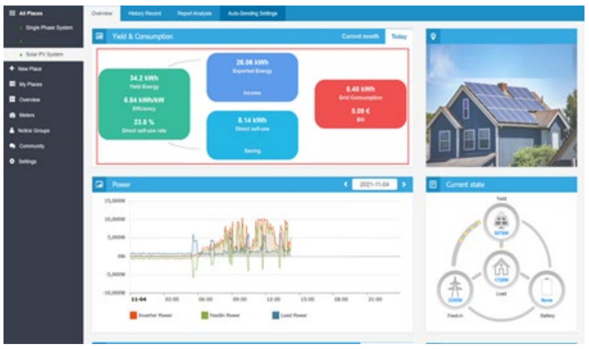


Figure 3: Django-based Web Dashboard

## 4.0 Experimental Setup and Data Collection

The experimental setup used for data collection is described in detail, along with the methodology employed to gather the energy consumption data necessary for training and evaluating the AI-based load prediction models.

### 4.1 Data Collection Process

Ensuring the relevance and accuracy of the load prediction models required the collection of a comprehensive dataset of energy consumption over a three-month period. Data were gathered locally using the developed edge-enabled energy meter system, with the PZEM-004T sensor connected to a specific section of the household electrical circuit within a controlled environment to monitor real-time energy usage. The Raspberry Pi, serving as the edge device, was configured to continuously read data from the sensor at fixed intervals of a few minutes. A data acquisition module parsed the raw sensor readings, extracted electrical parameters such as voltage, current, power, energy, and frequency, and appended a timestamp to each entry before storing the records in a local database on the Raspberry Pi. The three-month observation period was deliberately selected to capture a wide range of consumption patterns, including daily routines, weekly cycles, and potential monthly variations, thereby ensuring that the dataset would be representative enough to train robust predictive models.

### 4.2 Data Features

The quality and relevance of features are paramount for the performance of machine learning models. Based on common practices in load forecasting and the characteristics of energy consumption data, the following features were engineered and utilized for training the load prediction models:

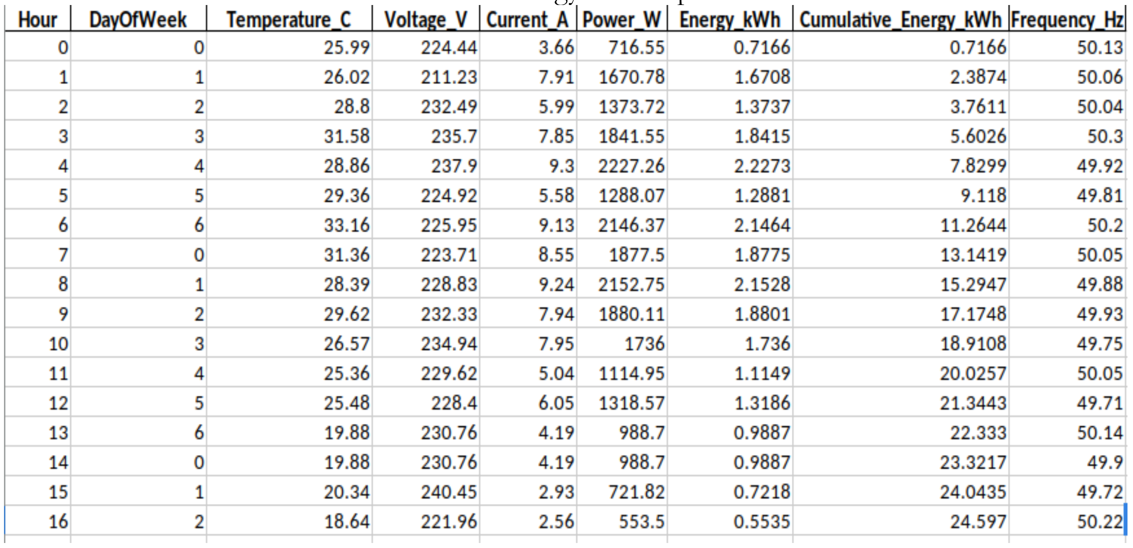
I. **Hour of the Day**: A categorical or numerical feature representing the hour (0–23) at which the energy consumption reading was taken. Energy consumption patterns typically show strong hourly periodicity, with distinct peaks and troughs corresponding to morning and evening activity.

II. **Day of the Week**: A categorical feature indicating the day (Monday to Sunday). Energy consumption often differs between weekdays and weekends due to variations in human activity and operational schedules.

III. **Historical Energy Consumption**: Lagged variables capturing past consumption values (e.g., previous hour, previous day, or same hour on the previous day). These features are critical for modeling temporal dependencies in energy data and improving time series forecasting performance.

Other potential features, such as humidity or specific device-level usage, were considered but not included in the initial model training. The focus remained on the core electrical parameters and temporal features. These additional factors could be explored in future work to further enhance prediction accuracy.

Table 1: A sample of the collected dataset, including hours of the day, days of the week, current, voltage, and historical energy consumption



| Hour | DayOfWeek | Temperature_C | Voltage_V | Current_A | Power_W | Energy_kWh | Cumulative_Energy_kWh | Frequency_Hz |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 25.99 | 224.44 | 3.66 | 716.55 | 0.7166 | 0.7166 | 50.13 |
| 1 | 1 | 26.02 | 211.23 | 7.91 | 1670.78 | 1.6708 | 2.3874 | 50.06 |
| 2 | 2 | 28.8 | 232.49 | 5.99 | 1373.72 | 1.3737 | 3.7611 | 50.04 |
| 3 | 3 | 31.58 | 235.7 | 7.85 | 1841.55 | 1.8415 | 5.6026 | 50.3 |
| 4 | 4 | 28.86 | 237.9 | 9.3 | 2227.26 | 2.2273 | 7.8299 | 49.92 |
| 5 | 5 | 29.36 | 224.92 | 5.58 | 1288.07 | 1.2881 | 9.118 | 49.81 |
| 6 | 6 | 33.16 | 225.95 | 9.13 | 2146.37 | 2.1464 | 11.2644 | 50.2 |
| 7 | 0 | 31.36 | 223.71 | 8.55 | 1877.5 | 1.8775 | 13.1419 | 50.05 |
| 8 | 1 | 28.39 | 228.83 | 9.24 | 2152.75 | 2.1528 | 15.2947 | 49.88 |
| 9 | 2 | 29.62 | 232.33 | 7.94 | 1880.11 | 1.8801 | 17.1748 | 49.93 |
| 10 | 3 | 26.57 | 234.94 | 7.95 | 1736 | 1.736 | 18.9108 | 49.75 |
| 11 | 4 | 25.36 | 229.62 | 5.04 | 1114.95 | 1.1149 | 20.0257 | 50.05 |
| 12 | 5 | 25.48 | 228.4 | 6.05 | 1318.57 | 1.3186 | 21.3443 | 49.71 |
| 13 | 6 | 19.88 | 230.76 | 4.19 | 988.7 | 0.9887 | 22.333 | 50.14 |
| 14 | 0 | 19.88 | 230.76 | 4.19 | 988.7 | 0.9887 | 23.3217 | 49.9 |
| 15 | 1 | 20.34 | 240.45 | 2.93 | 721.82 | 0.7218 | 24.0435 | 49.72 |
| 16 | 2 | 18.64 | 221.96 | 2.56 | 553.5 | 0.5535 | 24.597 | 50.22 |

Table 1 illustrates a sample of the collected dataset, including hours of the day, days of the week, current, voltage, and historical energy consumption.

### 4.3 Data Pre-Processing

Before feeding the collected data into the machine learning models, several preprocessing steps were undertaken to ensure data quality, consistency, and suitability for training. These steps were critical for preventing errors and optimizing model performance. First, missing data points, arising from sensor malfunctions or communication interruptions, were identified and addressed. For short gaps, linear interpolation was applied, while records with extensive missing values were removed. Second, outliers caused by sensor noise or unusual events were detected. If a clear correction pattern was identifiable, the values were adjusted; otherwise, the affected records were discarded to avoid skewing model performance. Third, continuous features (e.g., voltage, current, power, energy) were normalized or standardized to ensure they fell within a comparable range. This step was especially important for algorithms such as KNN and LSTM, which are sensitive to input scales, preventing larger-valued features from dominating the learning process. Fourth, categorical features such as Hour of the Day and Day of the Week were encoded into numerical form. One-hot encoding was applied to avoid imposing artificial ordinal relationships between categories. Finally, for time-series models like LSTM, the dataset was structured into sequential input–output pairs, allowing the model to capture temporal dependencies. For other algorithms (e.g., Linear Regression, Random Forest, KNN), lagged features were explicitly created to incorporate past consumption values as predictors. These preprocessing steps ensured that the dataset was clean, well-structured, and optimized for training, significantly enhancing the reliability and validity of the experimental results.

## 5.0 Load Prediction Model Development and Evaluation

This section details the development, training, and evaluation of the machine learning models used for load prediction. It covers the specific algorithms employed, the training methodology, the metrics used for performance assessment, and a comparative analysis of their results.

### 5.1 Machine Learning Models

Based on the literature review and the characteristics of the energy consumption data, four distinct machine learning models were selected for evaluation:

#### 5.1.1 Linear Regression

Linear Regression is a fundamental supervised learning algorithm used for modeling the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. In this context, it attempts to model the energy consumption as a linear combination of the input features (hour, day of the week, historical energy consumption). While simple and computationally inexpensive, its performance is contingent on the assumption of linearity in the underlying data relationships. It serves as a baseline model to assess the complexity required for accurate prediction.

#### 5.1.2 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm. For regression tasks, KNN predicts the value of a new data point based on the average of the values of its K nearest neighbors in the feature space. The 'distance' between data points is typically calculated using Euclidean distance. KNN is flexible and can capture non-linear relationships without making strong assumptions about the data distribution. However, its computational cost can increase significantly with larger datasets, and the choice of 'K' (number of neighbors) and distance metric can heavily influence its performance.

#### 5.1.3 Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mean prediction of the individual trees for regression tasks. Each tree in the forest is built from a random subset of the training data and a random subset of features, which helps to reduce overfitting and improve generalization. Random Forest is highly robust, capable of handling high-dimensional data, and can effectively capture complex non-linear relationships and interactions between features. Its inherent ability to provide feature importance also offers insights into the most influential factors affecting energy consumption.

#### 5.1.4 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a specialized type of recurrent neural network (RNN) designed to learn long-term dependencies, making them particularly suitable for time series forecasting. LSTMs address the vanishing gradient problem inherent in traditional RNNs through their unique gate mechanisms (input, forget, and output gates) that regulate the flow of information through the network. This allows LSTMs to selectively remember or forget information over extended sequences, which is crucial for capturing the temporal patterns and dependencies in energy consumption data that span hours, days, or even weeks.

### 5.2 Model Training

The collected and preprocessed three-month energy consumption dataset was first divided into training and testing sets, with 80% of the data allocated for model training and hyperparameter tuning, and the remaining 20% reserved as a hold-out set for final evaluation. Within the training portion, cross-validation techniques were employed to ensure robustness of model training and to mitigate overfitting. Each model was trained independently on the training dataset, with hyperparameter tuning performed for models such as KNN (optimizing the number of neighbors) and Random Forest (optimizing the number of trees, maximum depth, etc.) to achieve optimal performance. For the LSTM model, a sequential neural network architecture was designed, with parameters including the number of LSTM layers, units per layer, activation functions, and optimization algorithms adjusted over a specified number of epochs. This combination of cross-validation and a final hold-out test set ensured both reliable training and unbiased performance evaluation.

## 5.3 Evaluation Metrics

To quantitatively assess the performance of each load prediction model, three widely accepted regression evaluation metrics were used:

**Mean Absolute Error (MAE):** MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average of the absolute differences between predicted and actual values. A lower MAE indicates higher accuracy.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (1)$$

where yi is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations.

**Root Mean Square Error (RMSE):** RMSE is the square root of the average of the squared differences between predicted and actual values. It gives a relatively high weight to large errors, making it useful when large errors are particularly undesirable. A lower RMSE indicates higher accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (2)$$

where yi is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations.

**R² Score (Coefficient of Determination):** The R² score represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 1 indicates that the model perfectly predicts the target variable, and 0 indicates that the model explains none of the variability of the response data around its mean. A higher R² score indicates a better fit of the model to the data.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \quad (3)$$

Where yi is the actual value, $\hat{y}_i$ is the predicted value, $\bar{y}$ is the mean of the actual values, and n is the number of observations.

## 5.4 Model Performance and Computational Cost Comparison

The performance of four predictive models, Linear Regression, k-Nearest Neighbors (KNN), Random Forest, and a Long Short-Term Memory (LSTM) network, was rigorously evaluated using a 5-fold cross-validation approach to ensure robustness and generalizability across three months of locally collected energy meter data. Model performance was assessed using mean absolute error (MAE), root mean square error (RMSE), and the coefficient of determination (R²), while computational efficiency was evaluated through training time, inference time, and model size (Table 2). Among the models, Linear Regression achieved the best balance between accuracy and efficiency, with the lowest MAE (0.04104 ± 0.00199), lowest RMSE (0.05147 ± 0.00235), and highest R² (0.993 ± 0.00104). Notably, it also exhibited negligible computational overhead, with a training time of only 0.0018 s, an inference time of 0.000000424 s, and a compact model size of 0.05 MB, making it highly suitable for real-time deployment on resource-constrained edge devices. Random Forest also showed strong predictive accuracy (R² = 0.99019), but its training time was over 200 times longer than Linear Regression. KNN exhibited moderate predictive ability with higher inference latency, while the LSTM model underperformed substantially (R² = 0.60169) despite requiring the largest computational resources, including a 21.9 s training time and a 50 MB model size. These results reinforce that, for the target application, simple and lightweight models can outperform complex deep learning approaches in both accuracy and efficiency.

Table 2: Model Performance and Computational Cost Comparison

| Model | Avg. MAE ± Std | Avg. RMSE ± Std | Avg. R² ± Std | Training Time (s) | Inference Time (s) | Model Size (MB) |
|---|---|---|---|---|---|---|
| Linear Regression | 0.04104 ± 0.00199 | 0.05147 ± 0.00235 | 0.99299 ± 0.00104 | 0.001773 | 0.000000424 | 0.05 |
| K-Nearest Neighbors (KNN) | 0.13239 ± 0.01798 | 0.17399 ± 0.02768 | 0.91997 ± 0.02036 | 0.003430 | 0.000014116 | 5.00 |
| Random Forest | 0.04667 ± 0.00336 | 0.06107 ± 0.00844 | 0.99019 ± 0.00204 | 0.372304 | 0.000035200 | 10.00 |

| Model | Avg. MAE ± Std | Avg. RMSE ± Std | Avg. R² ± Std | Training Time (s) | Inference Time (s) | Model Size (MB) |
|---|---|---|---|---|---|---|
| LSTM Network | 0.29540 ± – | 0.39501 ± – | 0.60169 ± – | 21.857696 | 0.001359850 | 50.00 |

## 5.5 Feature Importance and Model Interpretability

A SHAP (SHapley Additive exPlanations) analysis was conducted to explain the factors influencing each model's predictions. This analysis highlighted the contribution of individual features to the model outputs, and the results were presented in a feature importance comparison bar chart in Figure 5, as well as in individual SHAP summary plots in Figure 6.

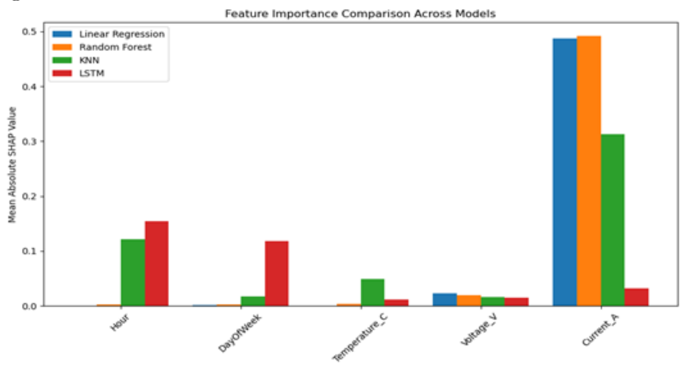


Figure 5: Feature importance comparison bar chart

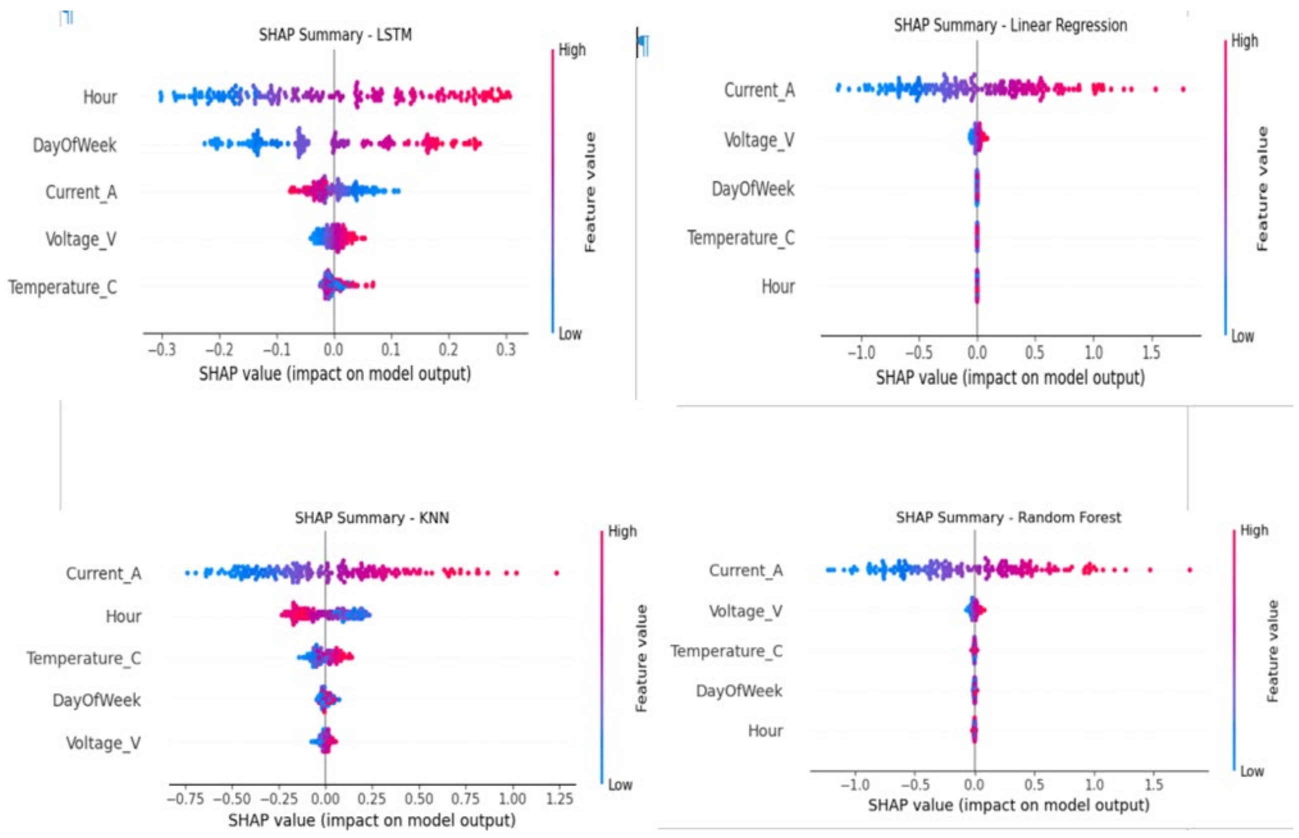


Figure 6: SHAP Summary Plots for LSTM, Linear Regression, KNN and Random Forest

The SHAP plots in figure 6 reveal a consistent pattern of feature importance across all models. For the Linear Regression model, Current_A and Voltage_V emerged as highly influential features. High values of Current_A drove a positive impact on the predicted output, while high values of Voltage_V had a negative impact. In conjunction with these electrical parameters, the temporal features, Hour and DayOfWeek, were also found to have a significant influence on predictions. This outcome is consistent with the model's ability to capture the complex interplay between real-time electrical parameters and predictable daily and weekly energy consumption patterns.

The Random Forest and KNN models demonstrated a similar reliance on both the electrical parameters and the temporal features as significant predictors. The consistency of these findings across three distinct model architectures strongly suggests that Current_A, Voltage_V, Hour, and DayOfWeek are the fundamental drivers of the target variable within this dataset.

In contrast, the LSTM network's feature importance was more broadly distributed. While Current_A and Voltage_V remained influential, the model's reliance on Hour and DayOfWeek was even more pronounced, as would be expected from a model designed for time-series analysis. However, as evidenced by its suboptimal performance, this reliance did not translate to improved final predictions, ultimately leading to a less effective and more computationally expensive model.

## 6.0 Discussion

The results of this study demonstrated that, despite its inherent simplicity, the Linear Regression model was the most effective solution for the given dataset. It outperformed all other benchmark models, including the more complex LSTM network, in both predictive performance and computational efficiency. This indicated that the relationship between the input features and the target variable, which included both electrical and temporal factors, was sufficiently linear that more advanced non-linear models such as Random Forest and LSTM were unnecessary.

The SHAP-based feature importance analysis provided a transparent view into each model's decision-making process, an essential step for building trust and explainability in AI-driven systems. The finding that load predictions were strongly influenced by electrical parameters (Current_A, Voltage_V) and temporal factors (Hour, DayOfWeek) validates the central premise of this research: a hybrid approach combining real-time measurements with historical context can yield accurate predictions. Although the LSTM model attempted to leverage temporal features, it did not achieve a corresponding improvement in accuracy. This outcome reinforced the conclusion that, for this dataset, the additional complexity and computational cost of deep learning models were not justified.

In practical terms, these findings have significant implications for real-world edge-enabled IoT applications. The superior accuracy, minimal computational footprint, and interpretability of the Linear Regression model make it highly suitable for resource-constrained environments such as the Raspberry Pi. Its negligible training and inference times allow real-time predictions with minimal latency, which is critical for effective energy management. Furthermore, as a transparent model, Linear Regression offers a distinct advantage in applications where user trust and regulatory compliance require explainability. By contrast, the high computational overhead of the LSTM network, without a corresponding gain in predictive power, highlights the limitations of deep learning approaches in edge computing environments.

The proposed solution therefore demonstrated a practical and transparent framework for residential and institutional energy management. Deploying the best-performing model on the Raspberry Pi and integrating it with a Django-powered dashboard for real-time monitoring and predictive analytics provided a complete, end-to-end solution. While the present study validated the effectiveness of Linear Regression in this context, future research could explore the impact of additional feature engineering techniques and the evaluation of the system on larger and more diverse datasets. Nonetheless, the evidence presented here confirms that Linear Regression remains a robust, efficient, and practical solution for AI-based load prediction in edge-enabled smart energy metering.

## 7.0 Conclusion and Recommendations/Future Work

### 7.1 Conclusion

This paper presented the design and evaluation of an edge-enabled IoT smart energy meter integrated with AI-based load prediction for device-level monitoring. The system employed a Raspberry Pi as the edge device and a PZEM-004T sensor for efficient local data acquisition, processing, and storage. Among the four evaluated machine learning models, Linear Regression achieved the best predictive accuracy ($R^2$: 0.993, MAE: 0.041, RMSE: 0.051), demonstrating that a simple, low-complexity model can outperform more computationally demanding approaches in this context. In contrast, the underperformance of the LSTM model underscored the practical limitations of deep learning in resource-constrained environments. K-fold cross-validation was employed to address potential data imbalance and improve generalizability, thereby enhancing the robustness of model evaluation. The deployment of the best-performing model on the Raspberry Pi, integrated with a Django-based web dashboard, resulted in a resource-efficient and scalable solution. Overall, the study reinforces the principle that lightweight, interpretable models, when properly validated and optimized, can deliver high predictive accuracy in edge-enabled IoT applications for both residential and institutional energy management.

### 7.2 Recommendation/Future Work

Future research should address the limitation of dataset diversity, as the present study relied on locally collected data that may not fully capture variations across geographical locations, environmental conditions, and usage patterns. Expanding the evaluation to larger and more heterogeneous datasets would enhance the robustness, adaptability, and reliability of the proposed solution in real-world deployments. Moreover, future work should emphasize validating the system under diverse operating conditions and investigating its scalability for larger-scale deployments, including integration with renewable energy sources and demand-side management frameworks. Such efforts will ensure the long-term adaptability, resilience, and effectiveness of edge-enabled IoT smart energy metering solutions for both residential and institutional applications.

## References

[1] Z. Nazari and P. Musilek, "Impact of Digital Transformation on the Energy Sector: A Review," Algorithms, vol. 16, no. 4, p. 211, 2023, doi: 10.3390/a16040211.

[2] Y. Yang, S. Xia, P. Huang and J. Qian, "Energy transition: Connotations, mechanisms and effects," Energy Strategy Reviews, vol. 52, p. 101320, 2024, doi: 10.1016/j.esr.2024.101320.

[3] B. Zohuri, "Navigating the Global Energy Landscape Balancing Growth, Demand, and Sustainability," Journal of Material Sciences & Applied Engineering, vol. 2, no. 4, pp. 01–07, 2023.

[4] K. Babu, S. Sivasubramanian, C. S. Nivetha, R. Senthil Kumar and M. soundari, "Intelligent energy management system for smart grids using machine learning algorithms," E3S Web of Conferences, vol. 387, p. 05004, 2023, doi: 10.1051/e3sconf/202338705004.

[5] P. Arévalo and F. Jurado, "Impact of Artificial Intelligence on the Planning and Operation of Distributed Energy Systems in Smart Grids," Energies, vol. 17, no. 17, p. 4501, 2024, doi: 10.3390/en17174501.

[6] N. H. Motlagh, M. Mohammadrezaei, J. Hunt and B. Zakeri, "Internet of Things (IoT) and the Energy Sector," Energies, vol. 13, no. 2, p. 494, 2020, doi: 10.3390/en13020494.

[7] A. Agarwal, S. P. Basu and S. Nayak, "IoT Data Management and Analytics: Challenges, Solutions & Trends," Journal of IoT in Social, Mobile, Analytics, and Cloud, vol. 5, no. 3, pp. 257–273, 2023, doi: 10.36548/jismac.2023.3.005.

[8] J. Huang, S. Zhou, G. Li and Q. Shen, "Real-time monitoring and optimization methods for user-side energy management based on edge computing," Scientific Reports, vol. 15, no. 1, p. 24890, 2025, doi: 10.1038/s41598-025-07592-4.

[9] D. N. Molokomme, A. J. Onumanyi and A. M. Abu-Mahfouz, "Edge Intelligence in Smart Grids: A Survey on Architectures, Offloading Models, Cyber Security Measures, and Challenges," Journal of Sensor and Actuator Networks, vol. 11, no. 3, p. 47, 2022, doi: 10.3390/jsan11030047.

[10] Y. Kang, S. Guo, P. Li and Y. Yang, "Edge Computing Based Privacy-Preserving Data Aggregation Scheme in Smart Grid," in 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), 2020, doi: 10.1109/IPCCC50635.2020.9391567.

[11] A. Saeed, R. M. Asif, A. U. Rehman, S. R. Hassan, S. Bharany and H. Hamam, "AI-Based Energy Management and Prediction System for Smart Cities," Jordan Journal of Electrical Engineering, vol. 11, no. 2, pp. 198–227, 2025, doi: 10.5455/jjee.204-1728140025.

[12] Y. M. Rind, M. H. Raza, M. Zubair, M. Q. Mehmood and Y. Massoud, "Smart Energy Meters for Smart Grids, an Internet of Things Perspective," Energies, vol. 16, no. 4, p. 1974, 2023, doi: 10.3390/en16041974.

[13] Z. Chen, A. M. Amani, X. Yu and M. Jalili, "Control and Optimisation of Power Grids Using Smart Meter Data: A Review," Sensors, vol. 23, no. 4, p. 2118, 2023, doi: 10.3390/s23042118.

[14] Y. Li, D. Qin, H. V. Poor and Y. Wang, "Introducing edge intelligence to smart meters via federated split learning," Nature Communications, vol. 15, no. 1, p. 9044, 2024, doi: 10.1038/s41467-024-53479-7.

[15] J. Kua, M. B. Hossain, I. Natgunanathan and Y. Xiang, "Privacy Preservation in Smart Meters: Current Status, Challenges and Future Directions," Sensors, vol. 23, no. 7, p. 3697, 2023, doi: 10.3390/s23073697.

[16] E. McKenna, I. Richardson and M. Thomson, "Smart meter data: Balancing consumer privacy concerns with legitimate applications," Energy Policy, vol. 41, pp. 807–814, 2012, doi: 10.1016/j.enpol.2011.11.049.

[17] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung and I.-H. Yen, "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes," Sensors, vol. 19, no. 9, p. 2047, 2019, doi: 10.3390/s19092047.

[18] N. Rodríguez-Pérez, J. Matanza Domingo and G. López López, "ICT Scalability and Replicability Analysis for Smart Grids: Methodology and Application," Energies, vol. 17, no. 3, p. 574, 2024, doi: 10.3390/en17030574.

[19] G. Hafeez, S. Y. Ahmad, K. Aurangzeb, K. Rehman, T. A. Khan and M. Alhussein, "A sustainable approach for demand side management considering demand response and renewable energy in smart grids," Frontiers in Energy Research, vol. 11, 2023, doi: 10.3389/fenrg.2023.1212304.

[20] K. Salameh, M. Awad, A. Makarfi, A.-H. Jallad and R. Chbeir, "Demand Side Management for Smart Houses: A Survey," Sustainability, vol. 13, no. 12, p. 6768, 2021, doi: 10.3390/su13126768.

[21] Z. Al-Waisi, M. Agyeman and M. El-Kishky, "On the Challenges and Opportunities of Smart Meters in Smart Homes and Smart Grids," in 2017 International Conference on Computer and Applications (ICCA), 2017.

[22] N. Uribe-Pérez, L. Hernández, D. De la Vega and I. Angulo, "State of the Art and Trends Review of Smart Metering in Electricity Grids," Applied Sciences, vol. 6, no. 3, p. 68, 2016, doi: 10.3390/app6030068.

[23] W. Fall, M. Badiane, P. A. A. Honadia and F. I. Barro, "Intelligent Energy Metering in the Smart Grid: A Review," International Journal of Applied Mathematics, Computational Science and Systems Engineering, vol. 6, pp. 173–185, 2024, doi: 10.37394/232026.2024.6.15.

[24] F. E. Abrahamsen, Y. Ai and M. Cheffena, "Communication Technologies for Smart Grid: A Comprehensive Survey," Sensors, vol. 21, no. 23, p. 8087, 2021, doi: 10.3390/s21238087.

[25] M. Pau, E. Patti, L. Barbierato, A. Estebsari, E. Pons, F. Ponci and A. Monti, "A cloud-based smart metering infrastructure for distribution grid services and automation," Sustainable Energy, Grids and Networks, vol. 15, pp. 14–25, 2018, doi: 10.1016/j.segan.2017.08.001.

[26] I. Sittón-Candanedo, R. S. Alonso, Ó. García, L. Muñoz and S. Rodríguez-González, "Edge Computing, IoT and Social Computing in Smart Energy Scenarios," Sensors, vol. 19, no. 15, p. 3353, 2019, doi: 10.3390/s19153353.

[27] Q. M. Nguyen, V.-H. Nguyen, V. K. Quy, L. A. Ngoc, A. Chehri and G. Jeon, "Edge Computing for IoT-Enabled Smart Grid: The Future of Energy," Energies, vol. 15, no. 17, p. 6140, 2022, doi: 10.3390/en15176140.

[28] W. Wang and Y. Li, "A comprehensive review of edge computing for power systems: State of the art, architecture, and applications," Applied Sciences, vol. 14, no. 8, p. 4592, 2024, doi: 10.3390/app14084592.

[29] I. Sittón-Candanedo et al., "Edge Computing, IoT and Social Computing in Smart Energy Scenarios," Sensors, vol. 19, no. 15, p. 3353, 2019, doi: 10.3390/s19153353.

[30] S. Redhu, A. Singh and B. A. Bremdal, "Edge computing for improving energy management in smart homes," in IET Conference Proceedings, 2023, pp. 11–14, doi: 10.1049/icp.2023.0815.

[31] A. Perçuku, D. Minkovska and N. Hinov, "Enhancing Electricity Load Forecasting with Machine Learning and Deep Learning," Informatics, vol. 11, no. 2, p. 59, 2024, doi: 10.3390/informatics11020059.

[32] M. W. Ahmad, M. Mourshed and Y. Rezgui, "An improved Linear Regression Model for Building Energy Consumption Prediction," in Procedia Computer Science, vol. 109, pp. 1121–1126, 2017, doi: 10.1016/j.procs.2017.05.385.

[33] O. Lahouar and J. Ben Hadj Slama, "Hour-ahead load forecasting using a K-nearest neighbor approach with wavelet transform," Energy Conversion and Management, vol. 91, pp. 312–320, 2015, doi: 10.1016/j.enconman.2014.12.037.

[34] R. D. Rai and J. Singh, "A review on Random Forest for short term load forecasting," in 2022 International Conference on Smart Grid and Renewable Energy (SGRE), 2022, pp. 1–5, doi: 10.1109/SGRE53750.2022.9863481.

[35] M. Abumohsen, A. Y. Owda and M. Owda, "Electrical Load Forecasting Using LSTM, GRU, and RNN Algorithms," Energies, vol. 16, no. 5, p. 2283, 2023, doi: 10.3390/en16052283.

[36] A. Unlu and M. Peña, "Comparative Analysis of Hybrid Deep Learning Models for Electricity Load Forecasting During Extreme Weather," Energies, vol. 18, no. 12, p. 3068, 2025, doi: 10.3390/en18123068.