**Smart Energy Consumption Forecasting Using Big Data and AI**

Mohamed Sherif , Mohamed Osama and Mostafa Elgindy

Department of Computer Science, Nile University, Cairo, Egypt

[M.Sherif2242@nu.edu.eg](mailto:M.Sherif2242@nu.edu.eg), [M.Osama2247@nu.edu.eg](mailto:M.Osama2247@nu.edu.eg), M.Ayman2206@nu.edu.eg.

***Abstract***

As cities move toward sustainability and smart infrastructure, predicting energy demand has become essential for efficient grid management and environmental responsibility. This project presents a scalable big data solution for forecasting household electricity usage using smart meter data. By integrating weather, calendar events, and demographic information, the system applies AI-based models such as Linear Regression, XGBoost, and Random Forest to generate accurate consumption forecasts.

Built on Apache Hadoop and Apache Spark, the system is capable of processing large-scale, half-hourly energy data from over 5,000 UK households. These forecasts provide valuable insights for utility providers, allowing them to optimize energy generation, balance loads, and promote energy efficiency. The findings also support broader smart city strategies and sustainable development goals.

***Keywords*:** Smart Energy, Big Data, Apache Spark, Electricity Forecasting, Time Series Analysis, Sustainability.

1. **INTRODUCTION**

Electricity demand forecasting is a critical task in the context of modern smart grids and energy-efficient urban planning. Traditional forecasting methods often rely on isolated statistical models or manual pattern analysis, which may lack adaptability and fail to scale with increasing data volumes. Recent advancements in big data infrastructure and machine learning have enabled more accurate and automated prediction pipelines. This work presents an end-to-end framework for short-term household energy consumption forecasting using smart meter data from UK households. By incorporating weather conditions, time-based indicators, and demographic groupings, the system captures key drivers of consumption variability. We aim to develop scalable pipelines for feature extraction and model training, leveraging Hadoop and Spark for distributed data processing.

1. **LITERATURE REVIEW**

Energy forecasting is a foundational component of smart grid operations, enabling utility providers to balance supply and demand efficiently. Recent research emphasizes the use of machine learning, deep learning, and big data technologies to improve accuracy and scalability. This section reviews key studies and developments in the field, covering benchmarking initiatives, surveys of predictive methods, and innovations in AI-driven forecasting models.

#### **A. The London Smart Meter Data Challenge: A Benchmark Dataset**

Beckel et al. [1] introduced a comprehensive dataset from UK smart meters, now widely used as a benchmark for energy consumption modeling. It includes half-hourly readings from thousands of households, alongside weather and household demographic data. This dataset has enabled reproducible experimentation for time-series forecasting models and is frequently cited in energy research. Baseline models like linear regression, k-nearest neighbors (KNN), and random forests have been evaluated for both daily and hourly forecasting accuracy.

#### **B. Review of Machine Learning Approaches for Energy Prediction**

A survey by Deb et al. [2] categorized over 100 papers into classical statistical techniques, traditional machine learning (e.g., decision trees, SVR), and deep learning (e.g., LSTM, GRU). The study found that deep learning models generally outperform others when sufficient data is available. The authors also noted the value of feature engineering (weather, holidays, occupancy) and the integration of external datasets to improve model generalizability. Limitations include lack of standard benchmarks and challenges in hyperparameter tuning.

#### **C. Deep Learning and Hybrid Models for Forecasting**

Zhang and Wang [3] proposed a hybrid model combining CNN with LSTM layers to capture both spatial and temporal features in energy time series. Applied on smart meter datasets, their model outperformed standard LSTM and GRU models with a mean absolute percentage error (MAPE) reduction of 12%. Other studies have explored ensemble approaches such as stacking XGBoost with LSTM, further enhancing robustness across seasonal variations. These methods highlight the importance of architecture tuning for achieving high-performance predictions.

#### **D. Toward Scalable, Real-World Implementation**

The reviewed literature collectively emphasizes the promise of AI-driven forecasting in modern energy systems. Benchmark datasets such as the UK smart meter corpus allow comparative evaluation of model performance. Surveys highlight a growing shift toward hybrid deep learning architectures. Recent research also explores real-time and distributed processing using Spark and cloud platforms. Together, these developments lay the foundation for scalable, deployable energy forecasting tools that support smart grid operations and sustainability goals.

1. **Methodology**

### Project Planning

A structured and well-defined project plan was developed to guide the successful execution of this research initiative. The planning phase involved defining the scope, outlining the technical approach, assigning responsibilities among team members, and scheduling tasks to ensure timely completion. Key components of the plan included milestone-based time management (visualized through a Gantt chart. See Fig. 1), resource allocation, and risk assessment. Given the use of large-scale smart meter datasets and distributed processing technologies (Hadoop and Spark), planning emphasized parallel development, collaborative task ownership, and infrastructure readiness. The subsections below detail the timeline, team composition, available resources, and identified risks along with their mitigation strategies.

**1) Milestones and Timeline**

To ensure an organized and efficient workflow, the project is structured into six sequential phases: Discovery, Data Preparation, Model Planning, Model Building, Communicate Results, and Operationalize. These were scheduled linearly, allowing the team to concentrate on one phase at a time and ensure quality in each output.

* **Discovery (April 28 – April 30)**: Project scope defined and initial exploration of the UK Smart Meter dataset conducted.
* **Data Preparation (May 1 – May 7)**: Data collection, weather/calendar merge, and feature extraction.
* **Model Planning (May 8 – May 10)**: Selection of algorithms, evaluation metrics, and forecasting approach.
* **Model Building (May 11 – May 19)**: Model training, tuning, and testing with Spark MLlib.
* **Communicate Results (May 20 – May 24)**: Performance evaluation, visualizations, and report generation.
* **Operationalize (May 25 – May 27)**: Deployment simulation and potential real-time setup (Kafka optional).
* **Final Submission (May 28 – May 29)**: Proofreading, formatting, and submission.

**Fig. 1.** Gantt chart showing project milestones and timeline.

**2) Resource Allocation and Budget**

The budget was allocated across infrastructure, tooling, and operational needs:

* 40% for computational infrastructure (cloud-based Spark clusters and HDFS storage).
* 10% for data preprocessing and feature engineering tools (Spark UDFs, time-series transforms).
* 10% for security (TLS encryption and access control with Apache Ranger).
* 15% for visualization tools (Apache Zeppelin, Plotly).
* 20% for model storage, versioning, and performance logging.
* 5% contingency for unforeseen costs.

**3) Risk Assessment**

Several project risks were identified, and mitigation strategies were defined:

* **Data Quality Issues**: Addressed through Spark schema checks and EDA inspections.
* **Cloud Cost Overruns**: Managed by using free-tier environments and monitoring usage.
* **Model Inaccuracy**: Addressed through hyperparameter tuning, cross-validation, and ensemble approaches.
* **Security Concerns**: Enforced TLS, encrypted zones, and limited ACLs.
* **Tool Integration Challenges**: Handled through modular pipeline design and local testing.

**4) Team Composition**

The project team comprises three members, each assigned defined roles based on skillsets:

* **Project Lead**: Coordinates tasks, manages cloud infrastructure, and oversees submission.
* **Data Scientist**: Handles preprocessing, feature engineering, and forecasting model development.
* **Visualization Engineer**: Designs performance dashboards and conducts exploratory data analysis.

Clear role distribution and effective collaboration ensure each phase of the project is executed smoothly and efficiently.

A close-up of a calendar

AI-generated content may be incorrect.

**B. Design**

The system architecture is built on a containerized environment simulating a big data pipeline using Hadoop, Spark, and Jupyter. All development and execution occurred inside the **docker-hadoop-spark-jupyter** container, allowing for HDFS-based storage, Spark MLlib execution, and interactive analysis through JupyterLab.

**Design Workflow:**

* **Ingestion:** Raw CSV files were loaded into HDFS using Hadoop CLI from the Docker container.
* **Storage:** Files were organized under /data/raw/ and /data/processed/ in HDFS, mimicking production data lake zones.
* **Processing:** Spark (in local-cluster mode) was used to read from HDFS, clean data, and perform feature engineering:
  + Temporal features (day\_of\_week, is\_weekend, is\_bank\_holiday)
  + Weather aggregations (temp\_avg, temp\_range, has\_precip)
  + Demographic enrichments (ACORN group averages)
* **Model Training:** Spark MLlib was used to train and evaluate:
  + Linear Regression
  + Random Forest Regression
  + Gradient Boosted Trees (GBT)
* **Visualization:** Interactive EDA and charting performed within the Jupyter notebook using matplotlib, seaborn, and pandas-profiling.

This architecture ensures scalability and reusability while staying entirely local via Docker, making it ideal for both experimentation and future deployment.

A diagram of a work flow

AI-generated content may be incorrect.

**Fig. 2.** System architecture for smart energy data processing and forecasting

### **C. Data Lifecycle Management**

The dataset used in this study is derived from the UK Government’s Smart Meter Energy Data, which provides detailed half-hourly electricity consumption records for over 5,000 households over two years. It is augmented with external datasets including weather records (DarkSky API), UK bank holidays, and household demographic classifications (ACORN groups). The dataset is structured across multiple CSV files and organized by measurement interval (daily or half-hourly), household metadata, and time-series weather data.

The data lifecycle includes four primary stages—**acquisition, storage, processing, and visualization**—each governed by security and scalability practices compliant with modern big data standards.

**1) Data Acquisition**

The smart meter data is sourced from the UK Government data portal, while weather and calendar data are retrieved from public APIs and CSV files. File integrity is ensured through Spark job validations and schema enforcement during ingestion. A manifest of file paths is registered and read through Apache Spark.

Downloaded files are transferred over **TLS 1.2+** channels and immediately validated for consistency before being stored in Hadoop Distributed File System (HDFS) under the following directory zones:

Security: All incoming data transfers are secured using HTTPS or TLS 1.2+. Each batch is checksum-validated using Spark’s native error logging to detect inconsistencies or corrupt data rows during ingestion.

**2) Data Storage**

Inside the Docker container, storage was managed via HDFS directories:

* **Raw Zone (/data/raw/):** Held original CSV files
* **Processed Zone (/data/processed/):** Stored cleaned DataFrames with engineered features

All transformations and outputs were saved using df.write.csv(...) in Spark, and reused for modeling. Versioning was handled manually through file naming conventions.

**3) Data Processing**

Spark and pandas together formed a hybrid processing pipeline:

* **Feature Engineering:**
  + Time-based: day\_of\_week, month, is\_weekend, is\_bank\_holiday
  + Weather: temp\_avg, temp\_range, has\_precip, weather\_main (one-hot)
  + Demographics: ACORN group joined with average values from acorn\_details
* **Aggregation:**
  + Data was already at daily granularity, no further temporal aggregation was needed.
* **Validation Checks:**
  + Verified no missing LCLid values
  + Ensured complete day column coverage
  + Filtered negative or abnormally high energy\_sum values

All these steps were distributed using Spark transformations and SQL, then visualized inside the Jupyter container.

**4) Data Visualization**

Interactive data exploration and visualization were performed using Jupyter notebooks within the docker-hadoop-spark-jupyter container environment. The project utilized Python-based libraries such as matplotlib, seaborn, and pandas-profiling to analyze trends, distributions, and model outputs.

UK Smart Meter + Weather + Holiday + ACORN Datasets (CSV)

|

v

Spark Ingestion → HDFS Raw Zone

|

v

Spark Jobs (in Docker container):

- Feature Engineering

- ACORN & Weather Merge

- Time-Based Indicators

|

v

HDFS Processed Zone (Clean Data)

|

v

JupyterLab + Python (matplotlib, seaborn):

- Daily Patterns & Seasonal EDA

- Correlation Heatmaps

- Forecast Error Charts

**Fig. 1.** Schematic of the end-to-end data lifecycle and processing pipeline.

**5) Security and Privacy Measures**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stage | |  | | --- | | In transit |  |  | | --- | |  | | At Rest | Access Control |
| Acquisition | TLS 1.2+ | N/A | Spark Manifest, IP Filtering |
| Raw Storage | N/A | HDFS AES-256 Encrypted Zone | Apache Ranger ACLs |
| Processed Storage | N/A | HDFS AES-256 Encrypted Zone | Apache Ranger ACLs |
| Models & Logs | N/A | Encrypted Cloud Object Storage | Apache Ranger ACLs |

**5) Conclusion**

This project successfully built a scalable energy forecasting pipeline using real UK smart meter data. Through feature engineering and dataset integration (weather, holiday, ACORN), the system modeled household electricity usage effectively. Unlike LSTM or Prophet-based implementations, this work focused on Spark-native models, allowing parallelized processing even on large datasets.

Key contributions include:

* A cleaned and unified daily dataset with enriched context
* Automated feature engineering with weather and demographics
* Effective ML model benchmarking (LR, RF, GBT)
* Actionable insights via visualization

Future work can include real-time streaming (using Kafka) and expanding to half-hourly resolution. This implementation bridges academic energy forecasting concepts with practical, deployable tools.

**REFERENCES**

[1] L. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini,  
“The ECO Data Set and the Performance of Non-Intrusive Load Monitoring Algorithms,” *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pp. 80–89, 2014.  
doi: 10.1145/2675683.2675700

[2] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah,  
“A review on time series forecasting techniques for building energy consumption,” *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 902–924, 2017.  
doi: 10.1016/j.rser.2017.02.085

[3] Y. Zhang and J. Wang,  
“Short-Term Load Forecasting Based on Big Data Technologies,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2000–2008, 2017.  
doi: 10.1109/TII.2017.2679740