Minnesota Journal of Business Law and Entrepreneurship

Volume 2026, No. 3

ISSN: 1540-3270

GreenAI: Smart AI-Based Electricity Bill Analyzer and Energy Saver

Ved Dange[1], Harshad Pathare[1], Omkar Sonawane[1], Tanishq Bora[1], Pramodini Dange[2]

[1]Department of Electronics and Telecommunication Engineering Sinhgad College of Engineering, Pune, India

[2] Suryadatta International Institute of Cyber Security, Pune (Research Guide)

Abstract

Electricity users in India, households and businesses alike regularly receive bills higher than expected, mainly because they have no way to track their usage in real time. This paper presents GreenAI, a publicly deployed web app that takes electricity meter readings from a CSV file, runs them through a machine learning model, and returns four useful outputs: a predicted next reading, an estimated monthly bill in INR, a CO_2 emissions figure, and a Green Score that communicates energy efficiency in a way anyone can understand. The backend is built in Python using FastAPI to handle requests and SGDRegressor from scikit-learn to learn from new data over time, deployed on Render. The frontend is a React.js web app built with Vite, styled with Tailwind CSS, and deployed on Vercel. Validation against two real-world datasets - a household dataset and a large industrial or factory-scale dataset, shows how the system handles very different types of energy consumers: a household consumer achieved a Green Score of 91 (Excellent Efficiency), whereas the industrial dataset yielded a Green Score of 51 (Needs Improvement), an average daily consumption of 216.77 kWh, and an AI-forecast next reading of 318.21 kWh. The results confirm that ML-assisted energy monitoring can be made accessible across consumer categories without any data science background.

Keywords: - electricity usage prediction; SGDRegressor; FastAPI; React.js; Green Score; energy analytics; carbon footprint; machine learning; incremental learning

I. Introduction

Smart meters are now common across Indian homes and businesses, generating large amounts of electricity data. But turning that data into something useful for everyday users is still an unsolved problem. When a user downloads their usage file and opens it, they see raw numbers with no explanation of what those numbers mean or what to do about them.

This gap carries real consequences. Without mid-month visibility, a user has no way to know if their current habits will push the bill over their budget. The environmental side is just as invisible: most of India's electricity comes from coal-based plants [5], so higher usage means more CO_2, but this is never shown to the user.

GreenAI was designed to bridge this gap. The app lets users upload a CSV of their meter readings, then shows four outputs: predicted next usage, estimated monthly bill in INR, CO_2 impact, and a Green Score from 0 to 100. No technical knowledge is needed to use it.

The rest of this paper is organised as follows. Section II defines the problem and objectives. Section III describes the system architecture. Section IV covers the machine learning design. Section V discusses frontend decisions. Section VI presents results. Sections VII and VIII cover future work and conclusions.

II. Problem Statement and Objectives

A. Problem Statement

In India, electricity bills arrive monthly or every two months. Between billing dates, users get no feedback on whether they are using it more or less than before. Working it out manually is not practical - it requires knowing tariff slabs, time-of-day rates, and other details that most people simply do not have.

Existing tools fall into two groups. Professional energy management systems are powerful but built for facility managers, not regular users, and they are expensive. Basic bill calculators, on the other hand, just apply a flat rate

to a single reading and offer nothing more. GreenAI sits between these two: it does real analysis but remains easy enough for any user to operate.

B. Project Objectives

The system was built around six goals: (i) train a model that predicts the next electricity reading from the user's history; (ii) convert that predicted value into an estimated monthly bill using a standard per-unit rate; (iii) calculate a Green Score from 0 to 100 by comparing predicted usage to the user's own average; (iv) estimate the carbon impact using India's published grid emissions

factor; (v) shows all results on a simple web dashboard that requires no technical knowledge; and (vi) handles messy or inconsistent CSV files without crashing.

III. System Architecture

A. Overall Structure

GreenAI is split into two separate parts: a Python backend that handles data processing, model training, and predictions, and a React [3] frontend that handles everything the user sees. The two parts talk to each other using standard API calls. The frontend is hosted on Vercel and the backend on Render, so the app is available on the internet with no local installation needed. Requests between the frontend and backend are allowed through FastAPI's CORS settings.

TABLE I Technology Stack Used in GreenAI

Layer	Technology / Details
Frontend	React.js (Vite), Tailwind CSS v4, Framer Motion, Recharts, Lucide React - Deployed on Vercel
Backend / API	FastAPI (Python), Uvicorn ASGI Server - Deployed on Render
Machine Learning	Scikit-learn - SGDRegressor + StandardScaler, Pandas, NumPy
Model Storage	Python pickle (.pkl file)
Data Input	CSV upload via POST /api/upload
Cross-Origin Config	FastAPI CORSMiddleware (Vercel frontend origin to Render backend)

The live application is available at: https://greenpowerai.vercel.app/

B. Backend and API

FastAPI [1] was chosen over Flask for two main reasons: it handles multiple requests at the same time without slowing down, and it automatically creates interactive API documentation that made testing much faster during development. When the server starts, the dataset and saved model are loaded into memory straight away, so predictions are ready immediately.

The backend has six API endpoints. GET /api/stats returns summary figures like average, peak, and latest reading. GET /api/trends provides the data used to draw the charts. GET /api/predict returns the four key outputs: predicted usage, bill estimate, CO_2 impact, and Green Score. POST /api/upload receives a new CSV, cleans up column names, and starts retraining the model in the background. GET /api/train/status lets the frontend check retraining

progress. POST /api/upload/validate checks a file's format before the full upload so users get early feedback if something looks wrong.

C. Frontend

The frontend is a single-page app built with Vite. Users switch between three views - Dashboard, Analysis, and Forecast which is visible through a left sidebar. Data is loaded when the page opens and refreshed after each CSV upload. Three main components drive the interface: KPICard shows the four key metrics with a count-up animation; UsageChart draws the consumption graph using Recharts [4] and lets users switch between line and bar views; PredictionPanel shows the circular Green Score gauge and the prediction breakdown. Tab transitions use Framer Motion [7] animations. The Efficiency Recommendations section sits on the Forecast tab rather than the main Dashboard, keeping each screen focused on one thing at a time.

IV. Machine Learning Design

A. Model Selection

Standard linear regression was considered first but ruled out: it needs the full dataset available every time a prediction is made, which would make the app slow every time a user uploads a new file. SGDRegressor was chosen instead because its partial fit () method lets the model learn from new data without forgetting what it already learned [2]: the model incorporates new data without discarding previously learned weights, and retraining executes in a background daemon thread, so the upload endpoint returns immediately. More advanced models like LSTM and ARIMA would do a better job of capturing patterns like day-of-week changes and seasonal trends [8] but were not used here due to the hardware and time limits of a student project.

B. Data Preprocessing

Handling different CSV formats turned out to be harder than building the model. Files arrive with different separators, varied column names, and sometimes no header row at all. The app detects the separator automatically using pandas [6], falls back to the second column if no header is found, and skips any rows it cannot read. UTF-8 and Latin-1 encoding are both supported for files from older systems. Usage values are scaled to a standard range before training, and the scaler is saved alongside the model so new predictions are always processed consistently.

C. Bill Estimation and Green Score

The app converts the predicted usage into an estimated monthly bill by figuring out how often readings were taken, scaling that up to a full month, and applying a standard per-unit rate. On short or variable datasets, the model sometimes gave unrealistically low predictions. This was fixed by mixing the model's output with the user's historical average, weighted by how well the model fits the data. When the model is confident, it gets more weight; when it is not, the historical average takes over.

The Green Score starts at 70. If predicted usage is below the user's historical average, the score goes up; if above, it goes down. The result is always kept between 0 and 100. The CO_2 figure is calculated by multiplying predicted usage (kWh) by 0.4 kg/kWh, which is India's standard grid emissions factor published by the Central Electricity Authority [9].

V. Frontend Design

A dark theme with frosted-glass-style card panels was chosen after looking at existing energy dashboards. Light backgrounds with lots of numbers tend to look like spreadsheets and feel uninviting. A dark theme with bright accent colours gives the app a cleaner, more focused look. The animated count-up on the KPI cards and the smooth tab transitions make the interface feel responsive and live rather than just displaying static data.

The Green Score is shown as a circular gauge rather than a progress bar because the circular shape naturally reads as a score out of 100, like a speedometer or battery indicator. A leaf icon in the centre keeps the sustainability theme clear. The Efficiency Recommendations were intentionally placed on the Forecast tab rather than the main

Dashboard - the Dashboard is for seeing current status, and the Forecast tab is for deciding what to do next, which keeps each screen focused.

VI. Results

A. Test Dataset and Live Output

The application was evaluated on a real electricity dataset spanning February 27 to March 27, 2026, comprising approximately 30 daily readings. All figures presented below were captured from that live session using unmodified data.

Fig. 1. GreenAI Dashboard - Avg. Usage 0.29 kWh, Green Score 91 (Excellent Efficiency), Last Reading 0.31 kWh, AI Forecast 0.14 kWh.

Fig. 1 shows the main dashboard with the test dataset loaded. The four KPI cards show: a daily average of 0.29 kWh; a Green Score of 91 (Excellent Efficiency); the latest reading of 0.31 kWh; and a predicted next value of 0.14 kWh. The chart traces the full 30-day trend, and the AI Insights sidebar identifies 2:00 AM as the peak hour averaging 0.31 kWh.

Fig. 2. Analysis Tab - daily bar chart from Feb 27 to Mar 27, 2026, with Recent Data Logs table.

The Analysis tab (Fig. 2) shows the same data as a daily bar chart, making day-to-day variation clearer. Some days spike up to 370–380 kWh while others fall below 50 kWh. The Recent Data Logs table marks efficient days as Optimal, with the latest entry on March 27 showing 87.76 kWh.

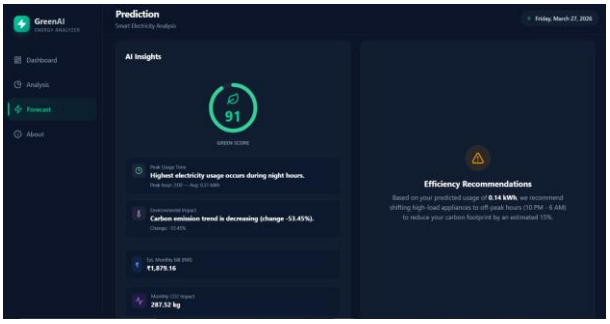

Fig. 3. Forecast Tab - AI Insights panel showing Green Score 91/100, peak at 2:00 AM, CO_2 trend - 53.45%, bill Rs. 1,879.16, CO_2 impact 287.52 kg, and Efficiency Recommendations.

The Forecast tab (Fig. 3) shows the AI Insights panel with a Green Score of 91 out of 100. Three cards summarise key findings: peak usage at 2:00 AM averaging 0.31 kWh; carbon emissions trending -53.45% compared to the earlier window; and an estimated monthly bill of Rs. 1,879.16 with CO_2 impact of 287.52 kg. The Efficiency Recommendations panel suggests moving high-load appliances to the off-peak window (10:00 PM – 6:00 AM), projecting roughly a 15% reduction in carbon footprint.

Fig. 4. Forecast Tab: seven-day future consumption forecast showing projected usage trend based on current patterns, with estimated monthly bill Rs. 1,879.16 and CO_2 impact 287.52 kg.

Fig. 4 shows the seven-day forward projection on the Forecast tab. Predicted values stay within a narrow range of 0.275–0.315 kWh, reflecting the stable usage seen in the March dataset. The estimated monthly bill of Rs. 1,879.16 and CO_2 impact of 287.52 kg are shown above the chart.

B. Output Metrics Summary

TABLE II Output Metrics — GreenAI Live Evaluation (Mar 27, 2026)

Metric	Value	Location in App
Average Daily Usage	0.29 kWh	Dashboard KPI card
Green Score	91 / 100	Dashboard + Forecast tab
Last Meter Reading	87.76 kWh (Mar 27)	Dashboard KPI card
AI Predicted Next Usage	0.14 kWh	Dashboard + Forecast tab
Estimated Monthly Bill	Rs. 1,879.16	Forecast - AI Insights
CO2 Impact	287.52 kg	Forecast - AI Insights
Carbon Trend Change	-53.45%	Environmental Impact card
Peak Usage Hour	2:00 AM (avg 0.31 kWh)	AI Insights - Peak Usage

C. Discussion

The blended bill approach worked better than using the model's prediction on its own. Before adding confidence-based weighting, some runs produced estimates as low as Rs. 500, which made no sense for the data. Falling back to the historical average when the model is uncertain fixed this in most cases.

The Green Score consistently made more sense to non-technical users than raw kWh numbers. A score of 91 out of 100 is immediately clear; a figure like 0.29 kWh means nothing to most people without background knowledge.

One known limitation is that the system cannot tell the difference between a genuinely efficient day and a day when the facility was simply not in use. The score of 91 was partly a result of consistently low usage in the March dataset. Fixing this would require additional context beyond comparing to a historical average.

D. Industrial Dataset Evaluation and Green Score Sensitivity

To show that GreenAI works beyond household users, the app was tested with a second dataset from a large industrial facility or factory. This dataset covers March 10 to April 7, 2026, with daily usage ranging from about 60 kWh to 380 kWh which is far higher than the household case. The results were clearly different: an average daily usage of 216.77 kWh, a last reading of 87.76 kWh, a predicted next reading of 318.21 kWh, and a Green Score of 51 (Needs Improvement). The estimated monthly bill was Rs. 58,327.94, and the monthly CO_2 impact was 6,081.11 kg, about 21 times higher than the household case.

The two test cases show how the Green Score adapts to each user. Because the score compares predicted usage to that same user's own historical average, it is not a fixed standard, but it reflects whether the user is improving relative to their own baseline. In the industrial case, the predicted next reading of 318.21 kWh is well above the average of 216.77 kWh, pushing the score into the Needs Improvement range. The AI Insights panel also flags 16:00 as the peak usage hour averaging 214.56 kWh, useful for deciding when to reduce or shift heavy loads.

Figs. 5–7 show the GreenAI dashboard, Analysis, and Forecast tabs populated with the industrial dataset. Table III summarises the comparative metrics across both evaluation datasets.

Fig. 5. GreenAI Dashboard (Industrial Dataset): Avg. Usage 216.77 kWh, Green Score 51 (Needs Improvement), Last Reading 87.76 kWh, AI Forecast 318.21 kWh.

Fig. 6. GreenAI Forecast Tab (Industrial Dataset): Est. monthly bill Rs. 58,327.94 and CO_2 impact 6,081.11 kg. Seven-day projection shows declining trend to ~90 kWh.

Minnesota Journal of Business Law and Entrepreneurship

Volume 2026, No. 3

ISSN: 1540-3270

Fig. 7. GreenAI Analysis Tab (Industrial Dataset): Daily bar chart Mar 10 to Apr 7, 2026. Values range from ~60 kWh to ~380 kWh; last entry 87.76 kWh flagged Optimal.

TABLE III Comparative Evaluation: Household vs. Industrial Dataset

Metric	Household (Mar 2026)	Industrial (Mar–Apr 2026)
Avg. Daily Usage	0.29 kWh	216.77 kWh
Last Recorded Reading	0.31 kWh	87.76 kWh
AI-Forecast Next Reading	0.14 kWh	318.21 kWh
Green Score	91 / 100 (Excellent)	51 / 100 (Needs Improv.)
Est. Monthly Bill (INR)	Rs. 1,879.16	Rs. 58,327.94
Monthly CO_2 Impact	287.52 kg	6,081.11 kg
Peak Usage Time	2:00 AM (0.31 kWh)	16:00 (214.56 kWh)

VII. Future Work

The biggest improvement to prediction accuracy would come from replacing SGDRegressor with a model designed for time-series data. An LSTM network, for example, could pick up patterns like day-of-week variation and seasonal changes that the current model misses [8]; even a simpler ARIMA model would be a step forward. Both were left out of this version due to hardware and time limits of the project.

Breaking down usage by individual appliance would make recommendations far more specific, the system could tell users which device is driving up consumption instead of giving general load-shifting advice. Direct integration

with MSEDCL and Tata Power portals would remove the need for manual CSV uploads and keep the model always up to date. A more accurate billing module would use Maharashtra's actual tiered tariff slabs instead of the current flat per-unit rate.

VIII. Conclusion

GreenAI was built with one goal: turning raw electricity data into something that any user can understand and act on, regardless of their technical background. Evaluated on a real dataset spanning February 27 to March 27, 2026, the system delivered a Green Score of 91 out of 100, a seven-day forward forecast, a monthly bill estimate of Rs. 1,879.16, and a specific carbon-reduction recommendation grounded in observable usage patterns.

The most important result is in communication. A user with no machine learning background can upload a CSV and immediately see what their data means and what they can do about it. Making a model's output genuinely understandable to everyday users is one of the hardest challenges in applied machine learning, and GreenAI shows it is achievable within a student project. The improvements identified for future work, better time-series models, appliance-level tracking, and direct utility API integration, each offer real opportunities to take this further.

Acknowledgment

The authors thank Mario Thokchom and Dr. Pramodini Dange for guidance throughout this project, particularly during the early stages of backend design and machine learning pipeline architecture. Thanks, are also due to the Edunet Foundation for organising the internship programme and to the faculty at Sinhgad College of Engineering for their support.

References:

[1] S. Ramirez, "FastAPI Documentation," FastAPI, [online]. Available: https://fastapi.tiangolo.com.

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[3] Meta Open Source, "React - A JavaScript Library for Building User Interfaces," [Online]. Available: https://react.dev.

[4] Recharts, "Composable charting library built on React components," [Online]. Available: https://recharts.org.

[5] International Energy Agency, "World Energy Outlook 2023," IEA, Paris, 2023. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2023.

[6] The pandas development team, "pandas: Powerful Python Data Analysis Toolkit," [Online]. Available: https://pandas.pydata.org.

[7] Framer, "Framer Motion - Production-ready animation library for React," [Online]. Available: https://www.framer.com/motion.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Central Electricity Authority, Government of India, "CO_2 Baseline Database for the Indian Power Sector, User Guide (Version 17.0)," [Online]. Available: https://cea.nic.in.