

GreenAI: Cloud-Based AI System for Carbon Emission Forecasting and GreenOps Optimization

Ananya K¹, Anusha L², Bindu A³, Preethi K R⁴, Dr. Hanumanthappa S⁵

^{1,2,3,4}UG Scholar, Department of ISE, Kalpataru Institute of Technology, Tiptur, Karnataka, India

⁵Professor, Department of ISE, Kalpataru Institute of Technology, Tiptur, Karnataka, India

Abstract

Cloud computing infrastructure is a growing contributor to global carbon emissions, yet most organizations have limited visibility into the environmental impact of their workloads. This work presents GreenAI, a full-stack cloud-based artificial intelligence (AI) sustainability platform that monitors cloud resource utilization, converts energy consumption into carbon emissions using region-specific grid carbon-intensity factors, forecasts future emissions using Random Forest and Extreme Gradient Boosting (XGBoost) models, and generates explainable rule-based GreenOps optimization recommendations. A background scheduler runs a daily monitoring cycle that forecasts next-day emissions, scores each earlier forecast against what actually occurred, detects statistical anomalies in server behavior, checks a configurable carbon budget, and dispatches Slack and e-mail notifications. A real-time streaming mode and an optional Amazon CloudWatch integration show that the architecture generalizes beyond simulated data. The backend is implemented in Python using FastAPI with MongoDB as the data store, and the frontend is a React single-page application with nine views. On a synthetic but realistically correlated dataset of 4,800 records (12 servers, 400 days), Random Forest and XGBoost achieved coefficient of determination (R^2) values of 0.9744 and 0.9738 with mean absolute errors of 2.813 and 2.852 kg of carbon dioxide (CO_2) on held-out data. The complete system passed an automated suite of 47 unit, integration, and application programming interface (API)-level tests. The demonstrated platform provides a foundation for cloud carbon accounting, emission forecasting, and sustainable cloud operations (GreenOps).

Keywords: Anomaly Detection, Carbon Footprint, Cloud Computing, GreenOps, Machine Learning, XGBoost

1. Introduction

Cloud computing has become a fundamental component of modern digital infrastructure, supporting applications that require large-scale computation, storage, and communication services. The increasing adoption of cloud-based applications has led to continuous growth in the number and capacity of data centres. Although virtualization and resource sharing improve infrastructure utilization, operating large computing facilities requires considerable electrical energy. The resulting environmental impact has made the forecasting and management of data-centre carbon emissions an increasingly important research area [12].

The environmental cost of computing cannot be represented by electricity consumption alone. The amount of carbon released for a given quantity of electricity depends on the composition of the electricity-generation mix and can change according to geographical region and time. Consequently, two workloads with similar computational requirements may produce different carbon footprints when executed under different energy conditions. Short-term forecasting of generation-based CO₂ emission factors has been demonstrated using time-series models [10], which supports a transition from conventional energy-efficient computing toward carbon-aware computing, in which environmental impact is incorporated into resource-management decisions.

An essential requirement for carbon-aware cloud management is reliable prediction of the emissions associated with computational activity. Statistical time-series models, particularly Autoregressive Integrated Moving Average (ARIMA), have been widely used for forecasting, including CO₂ emissions in India [1] and other economic and climate-related series [2], [6]. Machine-learning approaches have been used to predict CO₂ emission footprints [4], to compensate for insufficient local data by exploiting global patterns [3], and to reconstruct global daily emissions [11], while deep-learning models have been applied to traffic-related emissions [5]. However, reporting or forecasting emissions at an aggregate level provides limited support for preventing emissions in day-to-day operations. For operational sustainability, carbon information must be available early enough to influence resource-management decisions.

The increasing availability of cloud telemetry provides an opportunity to address this limitation through Artificial Intelligence (AI). Cloud platforms continuously generate information describing processor utilization, memory requirements, workload duration, network activity, and other infrastructure characteristics. These data can be analysed using machine-learning models to discover workload patterns and predict future resource requirements. Such predictions can provide an operational basis for adjusting infrastructure before changes in workload demand occur.

Forecasting further demonstrates the value of combining prediction with operational planning. Ostermann et al. [10] forecast generation-based CO₂ emission factors over short horizons using parametric and non-parametric time-series models, which provides advance information about the carbon intensity of electricity. Such information is useful because flexible workloads could be timed toward periods in which the expected carbon intensity is lower, even when the underlying workload remains unchanged.

Another related direction is the forecasting of emissions specifically for data-centre infrastructure. Jha et al. [12] applied AI models to forecast CO₂ emissions of data centres in the United States and discussed emission-reduction strategies and policy recommendations. This work confirms that AI-based forecasting is applicable to data-centre emissions. Operational cloud management additionally requires predictions at the level of individual servers and regions, which is the level targeted by the proposed framework.

A further body of work compares families of forecasting models. Ajala et al. [7] compared machine-learning, deep-learning [14], and statistical models for daily CO₂ emission prediction, Luo [8] examined Prophet, ARIMA, and Long Short-Term Memory (LSTM) models, and Wen et al. [9] proposed a hybrid ARIMA-LSTM model for emissions in China and its regions. These studies suggest that the choice of forecasting model should be validated empirically for each application rather than assumed in advance.

Forecasting therefore represents a central requirement for the development of intelligent Green Operations. Future cloud demand can fluctuate substantially, and electricity carbon intensity can also vary over time. A system that responds only to current measurements may miss opportunities for proactive optimization. In contrast, forecasting can provide advance information that enables resource scaling,

workload redistribution, scheduling adjustments, and resource deallocation to be performed before inefficient operating conditions occur.

This study proposes an AI-Based Carbon Emission Forecasting and Green Ops Optimization framework for cloud computing environments. The proposed framework uses cloud workload telemetry, including CPU utilization, memory allocation, execution duration, and network throughput, to model workload behaviour and estimate future carbon emissions. The forecasting layer provides information to an optimization component that identifies opportunities for improving resource utilization while reducing unnecessary environmental impact.

The proposed framework is designed around a continuous operational cycle in which cloud measurements are collected, processed, analysed, and converted into predictive information. The resulting predictions are then used to support Green Ops decisions such as dynamic resource scaling, workload redistribution, scheduling changes, and deallocation of idle resources. In this manner, the proposed system aims to transform carbon monitoring from a reporting-oriented activity into a predictive decision-support mechanism.

The principal contributions of this research are the integration of AI-based carbon-emission forecasting with cloud resource optimization, the use of workload telemetry for predictive environmental analysis, and the development of a Green Ops decision layer capable of translating predictions into resource-management actions. The framework is intended to provide a scalable foundation for incorporating carbon considerations into routine cloud operations without treating sustainability as an independent post-processing activity.

2. Literature Survey

Research on carbon-emission forecasting has progressed from classical statistical time-series models toward machine-learning, deep-learning, and hybrid approaches, and more recently toward forecasting at the level of electricity generation and data centres. Together, these studies provide the technical foundations for AI-driven Green Operations. This section reviews them according to model family and application scale.

Statistical time-series models form the earliest line of work. H. P. M. et al. [1] used ARIMA to forecast CO₂ emissions in India, demonstrating that classical time-series analysis can be applied to national emission data. Dar et al. [2] applied ARIMA to historical stock data to produce future projections, which illustrates the generality of the method beyond environmental data. Fan [6] compared ARIMA and Exponential Smoothing (ETS) models for global temperature-anomaly forecasting. Such models are interpretable and effective for single series, but they typically model one variable at a time and do not directly incorporate multivariate resource-utilization features such as CPU, memory, or network activity. Machine-learning techniques have been applied to emission prediction in several settings. Meng and Hossain [4] used machine learning to predict CO₂ emission footprints, while Hung [3] proposed a machine-learning approach that leverages global patterns to overcome data insufficiency and improve model accuracy. Li et al. [11] used machine learning to reconstruct global daily CO₂ emissions. These studies show that machine-learning models can capture nonlinear relationships between explanatory variables and emissions. The treatment of data insufficiency in [3] is also relevant to the present work, because production cloud telemetry is often unavailable for model development.

Deep-learning and hybrid models extend this direction. Al-Nefaie and Aldhyani [5] used a deep-learning model to predict CO₂ emissions from traffic vehicles for a sustainable and smart environment. Luo [8]

compared Prophet, ARIMA, and LSTM models for CO₂ emission prediction, and Wen et al. [9] proposed a hybrid ARIMA-LSTM model for China and its regions, combining a statistical component with a neural sequence model. These works illustrate that deep and hybrid models can represent nonlinear temporal dependence, although they may require larger data volumes and offer less transparency than tree-based or rule-based approaches.

Comparative studies are particularly relevant to model selection. Ajala et al. [7] compared machine-learning, deep-learning, and statistical models for daily CO₂ emission prediction. Daily resolution corresponds closely to the daily monitoring cycle of the proposed framework, and the comparative design suggests that competing models should be evaluated side by side on the same data. The proposed framework follows this practice by training two independent regressors and comparing them on held-out data.

Forecasting of electricity carbon intensity is another important direction. Ostermann et al. [10] forecast short-term generation-based CO₂ emission factors for Germany using parametric and non-parametric time-series models. This work is significant because the emissions of a workload depend on both the energy it consumes and the carbon intensity of the electricity supply, and both can vary over time. It supports the extension of the proposed framework from static regional carbon-intensity factors toward live or forecast carbon-intensity inputs.

The forecasting of emissions for data-centre infrastructure is the most directly related direction. Jha et al. [12] forecast CO₂ emissions of data centres in the United States using AI models and discussed emission-reduction strategies and policy recommendations. This work links AI-based forecasting to data-centre emissions and to mitigation. The present work complements it by operating on server-level and region-level telemetry and by converting forecasts into resource-level recommendations [13].

Despite substantial progress, the literature reveals an integration gap. The reviewed studies [1]–[12] concentrate primarily on forecasting emissions or related time series and are evaluated mainly in terms of predictive accuracy. Statistical, machine-learning, deep-learning, and hybrid models have each been shown to be useful, but workload prediction, carbon-emission forecasting, and resource optimization are not consistently combined within a single operational framework. Furthermore, forecasts are rarely connected to concrete resource-management actions such as scaling, redistribution, scheduling, or deallocation.

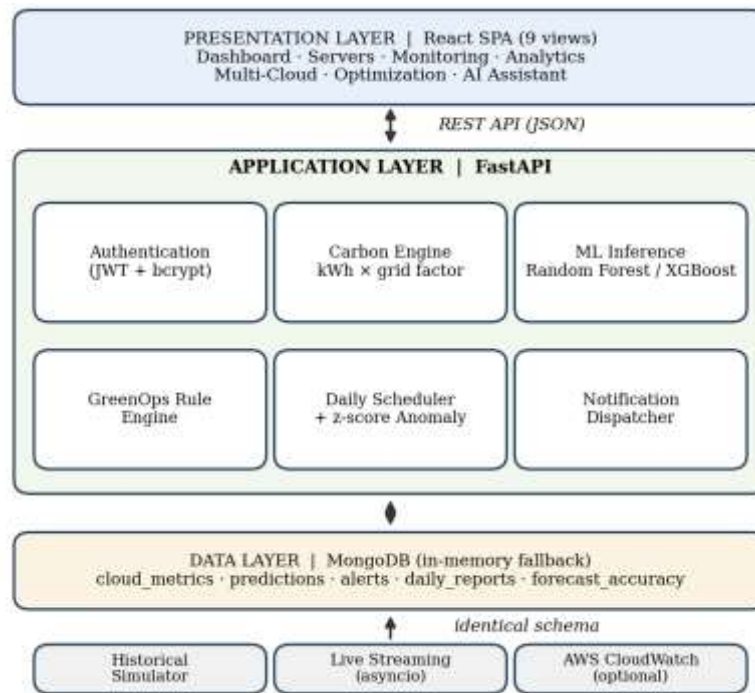
The proposed AI-Based Carbon Emission Forecasting and Green Ops Optimization framework is intended to address this gap by connecting these functions through a continuous decision process. Cloud workload telemetry is first processed to obtain relevant resource indicators. AI models then use these indicators to forecast workload behaviour and associated carbon emissions. The resulting information is supplied to the Green Ops optimization layer, which can identify opportunities for resource scaling, workload redistribution, scheduling adjustments, and idle-resource deallocation.

Consequently, the proposed research moves beyond the conventional separation between carbon monitoring and cloud resource management. Its central premise is that carbon information becomes more useful when it is incorporated into predictive operational decisions. By connecting cloud telemetry, AI forecasting, carbon estimation, and resource optimization, the framework provides an integrated approach for reducing avoidable resource consumption and supporting environmentally responsible cloud operations.

3. Methodology

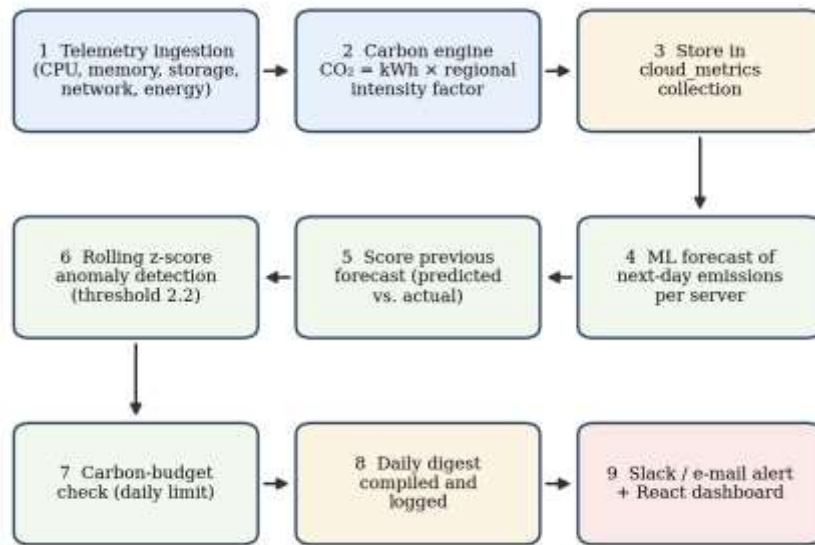
GreenAI follows a layered, three-tier architecture. The presentation layer is a React single-page application that communicates exclusively through a Representational State Transfer (REST) application programming interface (API). The application layer, built with FastAPI, hosts authentication, telemetry ingestion, the carbon calculation engine, the machine-learning inference layer, the GreenOps recommendation engine, the daily monitoring scheduler, and the notification dispatcher. The data layer is MongoDB, storing users, telemetry, predictions, alerts, daily digests, and configuration.

Figure 1: Layered Three-Tier Architecture of the GreenAI System



A distinguishing design decision is that all three telemetry sources, namely a one-shot historical simulator, a real-time asynchronous streaming engine, and an optional Amazon CloudWatch integration, write into an identical schema. Every downstream component therefore operates identically regardless of data provenance, so the prototype can move toward production infrastructure without architectural change. The system uses only open-source components and degrades gracefully to an in-memory database when MongoDB is unavailable. The daily monitoring cycle forecasts next-day emissions, scores the previous forecast, detects anomalies, checks the carbon budget, and compiles a digest. Any alert is dispatched to Slack and/or e-mail and is logged regardless of delivery outcome. All results are exposed through REST endpoints and rendered by the dashboard as charts, tables, and cards.

Figure 2: Data Flow from Telemetry Ingestion to Daily Monitoring, Alerts, and Dashboard



3.1 Dataset and Data Acquisition

Because production cloud telemetry was not available, a synthetic but realistically correlated dataset was generated to train and evaluate the forecasting models. The generator simulates 12 servers across 6 cloud regions over 400 days, with one reading per server per day. It builds in weekly seasonality (lower weekend utilization), a slow long-term upward trend, energy consumption computed as a weighted function of central processing unit (CPU), memory, storage, and network usage, and carbon emission computed as energy multiplied by the regional carbon-intensity factor with a small proportional noise term. A lag feature (previous day's emission) captures day-to-day autocorrelation. Dataset properties are summarized in Table 1.

Table 1: Dataset Specification

Property	Value
Simulated servers	12
Cloud regions	6 (ap-south-1, us-east-1, us-west-2, eu-west-1, eu-north-1, sa-east-1)
Time span	400 days per server
Total records	4,800 rows
Input features	CPU, memory, storage, network, temperature, energy, region, previous-day emission
Target variable	carbon_emission (kg CO ₂)
Train / test split	80% / 20%, fixed random state

3.2 Carbon Calculation Engine

The carbon engine implements the standard emissions-accounting relationship used by open-source estimation tools, in which energy consumed is multiplied by a regional carbon-intensity factor:

$$CE \text{ (kg CO}_2\text{)} = E \text{ (kWh)} \times CI \text{ (kg CO}_2\text{/kWh)}$$

Here, CE is the carbon emission, E is the energy consumed, and CI is the regional carbon-intensity factor. Region-specific factors reflect real differences in grid composition, so a coal-heavy grid produces substantially more emissions per kWh than a hydro- or nuclear-dominant grid. The factors are currently static; a live source such as Electricity Maps is proposed as an extension. A composite Sustainability (Green) Score from 0 to 100 combines utilization efficiency, which penalizes both idle waste and overload, with carbon efficiency.

3.3 Machine-Learning Forecasting

The forecasting pipeline follows a standard supervised-learning workflow: data loading, cleaning of missing or out-of-bound records, feature engineering (label-encoding of region and inclusion of the lag feature), an 80/20 train-test split, and training of two independent regressors. These are a Random Forest Regressor with 250 estimators, implemented in scikit-learn, and an XGBoost Regressor with 300 boosting rounds. Models are evaluated using mean absolute error (MAE) and the coefficient of determination (R^2), and the trained models, label encoder, and metrics are persisted for the inference layer.

3.4 GreenOps Recommendation Engine

Instead of an opaque classifier, the recommendation engine applies a transparent rule set to each server's recent telemetry so that recommendations can be justified to engineering and finance stakeholders. Five rules are implemented: Resource Optimization (average CPU below 20%: downsize or consolidate), Workload Optimization (average CPU or memory above 88%: enable autoscaling or distribute load), Green Region Migration (high energy use in a carbon-intensive region: migrate non-latency-sensitive workloads), Storage Cleanup (high storage with low compute: archive cold data and remove unused snapshots), and Workload Scheduling (elevated energy use: shift batch jobs to lower-carbon hours).

3.5 Daily Monitoring and Anomaly Detection

The daily cycle runs on a 24-hour schedule using APScheduler, or on demand for demonstration. It forecasts next-day emissions for every active server, scores the previous forecast against the readings that have since arrived, runs anomaly detection, checks the total against a configurable carbon budget, and compiles a digest. Anomaly detection uses a rolling z-score: for each server, the mean and standard deviation of recent emissions are computed, and the latest reading is flagged if it deviates by more than a configurable threshold (default 2.2 standard deviations). This choice makes each alert explainable in plain terms, which supports operator trust.

4. Experimentation

The backend uses Python 3.12 with FastAPI for the REST API, MongoDB 7 for persistence (with an automatic in-memory fallback), JSON Web Token (JWT) with bcrypt for stateless authentication and password hashing, scikit-learn and XGBoost for forecasting, and APScheduler with asyncio for background processing. Notifications are sent through Simple Mail Transfer Protocol (SMTP) e-mail and Slack incoming webhooks, and the optional real-data connector uses boto3 to read Amazon CloudWatch metrics. The frontend uses React 18 with Vite, Tailwind CSS, Recharts, and Axios, and the system can be deployed with Docker Compose. The minimum hardware is a dual-core 2.0 GHz processor, 4 GB random access memory (RAM) (8 GB recommended for model training), and 500 MB of storage.

A real-time Live Mode inserts one new telemetry reading per server every few seconds using an asyncio task, with CPU and memory following a random walk so that the stream drifts organically. Every alert dispatch is logged with its success or failure status, and a failed notification never interrupts the monitoring cycle. An AI Assistant applies a lightweight retrieval-augmented pattern: it retrieves a server's recent

telemetry as grounding context and generates answers with a deterministic template engine, with the retrieval path designed to accept a large-language-model (LLM) backend.

The dashboard comprises nine views: Login, Dashboard, Servers with a drill-down page, Monitoring, Analytics, Multi-Cloud, Optimization, and AI Assistant. The demonstrated outputs include live KPIs, emission trends, forecasts with model confidence, per-server recommendations, and the conversational assistant, with loading skeletons, empty states, and retryable error banners.

5. Results and Discussion

The system was validated with a multi-layered Pytest strategy using FastAPI's TestClient: unit tests for the carbon engine, GreenOps rules, and anomaly detector; integration tests of full API request/response cycles against an in-memory database; and edge-case tests for invalid credentials, missing servers, misconfigured notification channels, and division-by-zero conditions. The suite comprises 47 test cases across seven modules, and all 47 pass. Representative cases are listed in Table 2.

Both models were evaluated on the held-out 20% test split. Random Forest achieved an R^2 of 0.9744 with an MAE of 2.813 kg CO₂, and XGBoost achieved an R^2 of 0.9738 with an MAE of 2.852 kg CO₂ (Table 3). The comparable, high accuracy confirms that utilization metrics, region, and the lag feature carry strong predictive signal. Random Forest was selected as the default model because of its marginally higher R^2 and lower error, and XGBoost remains available as a user-selectable alternative.

Table 2: Representative Test Cases

Test Area	Scenario	Observed Outcome
Carbon engine	500 kWh at India grid factor (0.82)	410.0 kg CO ₂ (Pass)
GreenOps engine	Idle (CPU < 20%) and overloaded (CPU > 88%) telemetry	Resource and Workload Optimization recommendations fire (Pass)
Anomaly detection	Spike above a stable or zero-variance baseline	Anomaly flagged for the correct server (Pass)
Authentication	Duplicate e-mail; unauthenticated request	Hypertext Transfer Protocol (HTTP) 409 and HTTP 401 returned (Pass)
Daily monitoring	Two consecutive runs; budget set below usage	Accuracy record created on second run; budget alert raised (Pass)
Notifications	Alert dispatch with a failing webhook	Daily job completes regardless (Pass)

Table 3: Forecasting Model Performance

Model	R^2 Score	MAE (kg CO ₂)
Random Forest (250 estimators)	0.9744	2.813
XGBoost (300 boosting rounds)	0.9738	2.852

The deployed system was also verified end-to-end on a live instance. A full daily cycle forecast emissions for six servers, scored forecast accuracy on the second consecutive run, detected an injected anomaly with a z-score of about 82, and raised a budget-breach alert when the limit was set below actual usage. Live Mode produced four new database records over a ten-second interval under simultaneous request load,

and a failing webhook was handled without interrupting the cycle. Representative screens are shown in Figures 3 to 6.

Figure 3: GreenOps Dashboard: Live KPIs, Carbon-Emission Trend, and Sustainability Score

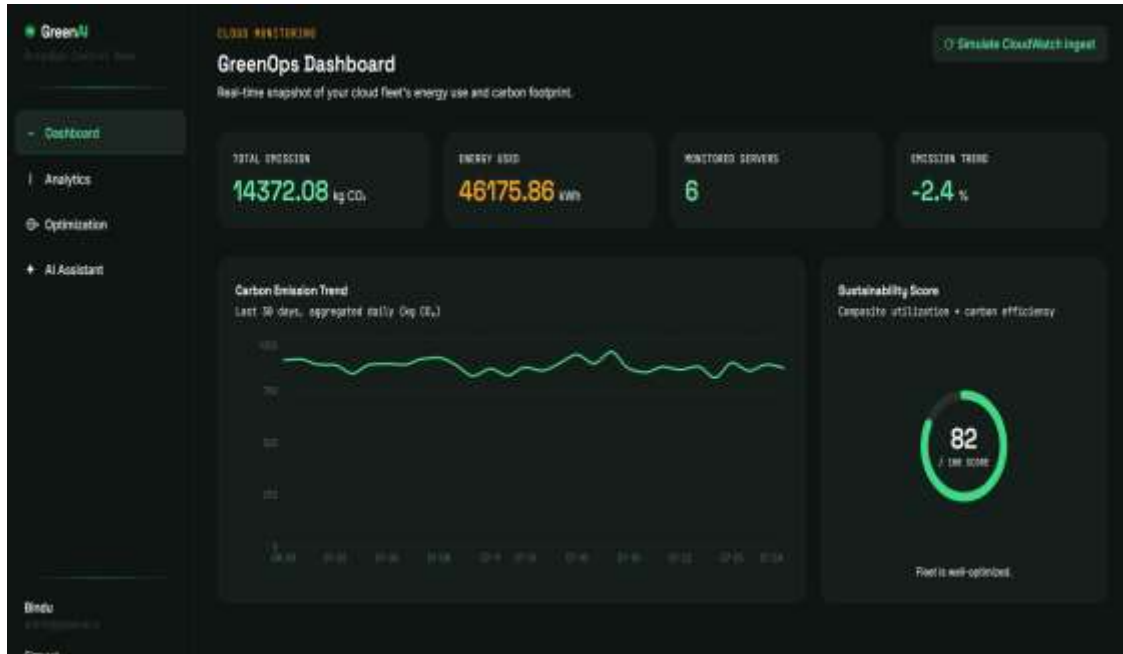


Figure 4: Analytics View: ML-Based Emission Forecasting with Model Confidence

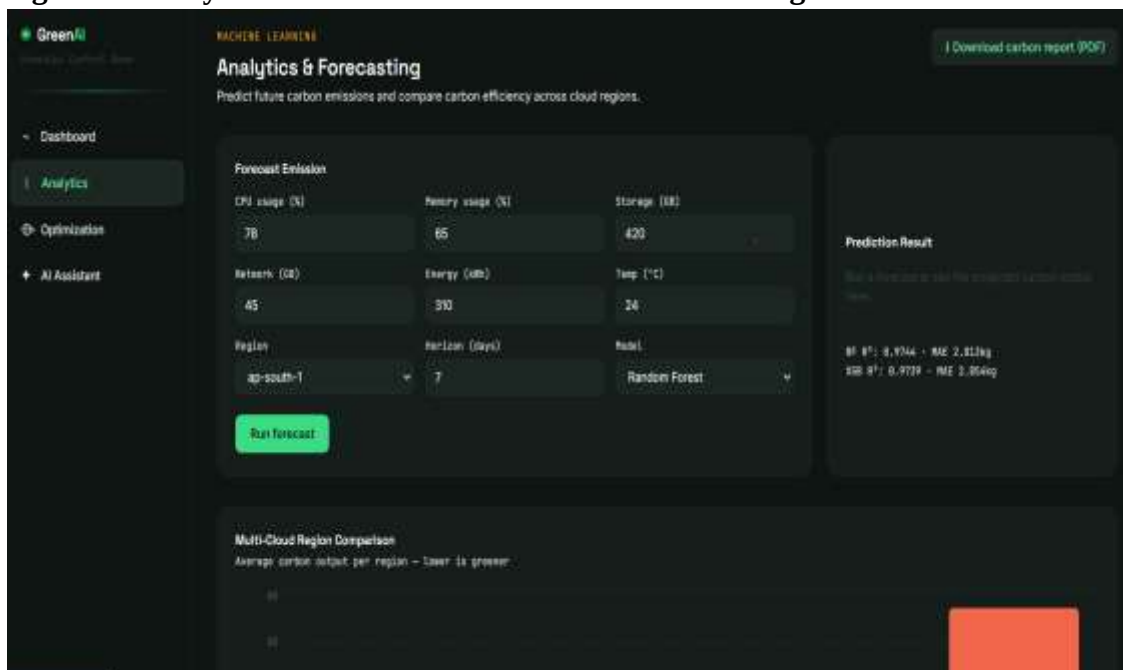


Figure 5: Optimization View: Rule-Based GreenOps Recommendations per Server

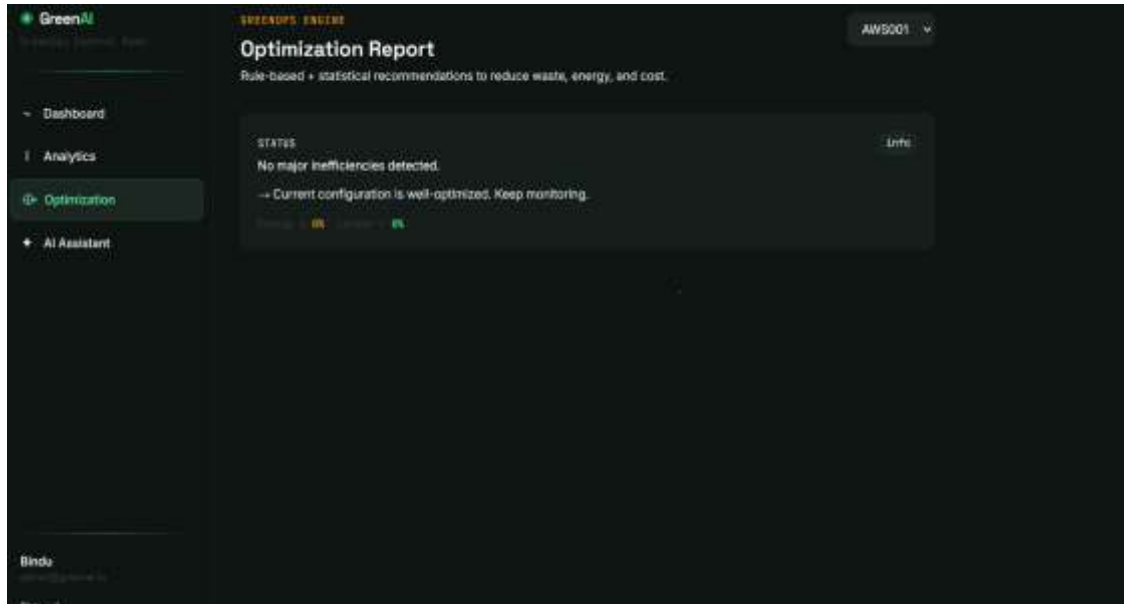
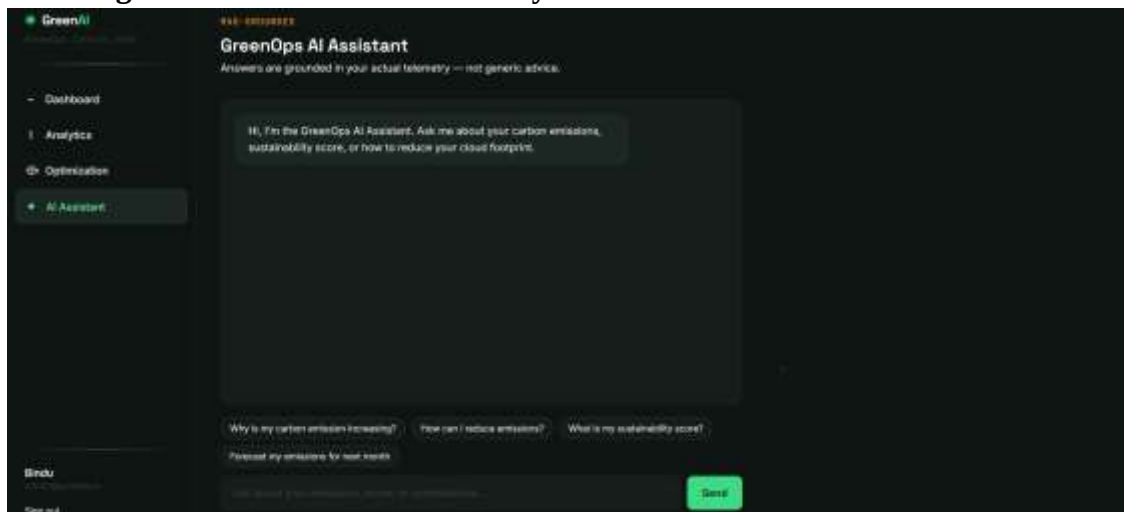


Figure 6: AI Assistant: Telemetry-Grounded Conversational Interface



The main limitations are that the training dataset is synthetic rather than production telemetry, that regional carbon-intensity factors are static rather than a live feed, that the default AI Assistant is rule-based rather than LLM-driven, that the CloudWatch integration currently retrieves CPU utilization only, and that the system has not been load-tested at production scale.

6. Conclusion

This work designed, implemented, and validated GreenAI, an integrated cloud sustainability platform that combines resource monitoring, region-aware carbon accounting, machine-learning emission forecasting, an explainable recommendation engine, and an automated daily monitoring cycle with anomaly detection, carbon-budget alerting, and Slack and e-mail notifications. Unlike descriptive-only carbon dashboards, the system scores its own forecasts against verified outcomes, and it supports real-time data through a live-streaming engine and an optional Amazon CloudWatch integration. The forecasting models achieved

$R^2 \approx 0.97$, and the system passed a 47-case automated test suite.

The next development stage should integrate a live grid carbon-intensity API such as Electricity Maps, connect the AI Assistant to a production large language model, extend the CloudWatch connector to memory, disk, and network metrics and add Azure and Google Cloud connectors, incorporate the scaffolded Long Short-Term Memory (LSTM) model as a third forecasting option, introduce role-based access control, add a Slack slash-command interface, and conduct load and scalability testing.

7. References

1. H. P. M., M. Z. Rehman, A. A. Dar, and T. W. Wangmo, "Forecasting CO₂ Emissions in India: A Time-Series Analysis Using ARIMA," *Processes*, vol. 12, no. 12, Art. 2699, 2024.
2. A. A. Dar, A. Jain, and M. Malhotra, "Time Series Analysis with ARIMA for Historical Stock Data and Future Projections," *Soft Computing*, vol. 28, pp. 12531–12542, 2024.
3. G. Hung, "Advanced Machine Learning Approach for CO₂ Emission Forecasting: Leveraging Global Patterns to Overcome Data Insufficiency and Improve Model Accuracy," *Journal of Human & Social Sciences*, vol. 9, no. 1, 2025.
4. Y. Meng and N. Hossain, "Predicting CO₂ Emission Footprint Using AI through Machine Learning," *Atmosphere*, vol. 13, no. 11, Art. 1871, 2022.
5. A. H. Al-Nefaie and T. H. Aldhyani, "Predicting CO₂ Emissions from Traffic Vehicles for Sustainable and Smart Environment Using a Deep Learning Model," *Sustainability*, vol. 15, no. 9, Art. 7615, 2023.
6. C. Fan, "Global Temperature Anomaly Forecast: A Comparative Analysis of ARIMA and ETS Models," *Proceedings of the Asia-Pacific Environment Management and Policy Summit (AEMPS)*, 2023.
7. A. A. Ajala, O. L. Adeoye, and O. M. Salami, "An Examination of Daily CO₂ Emissions Prediction through a Comparative Analysis of Machine Learning, Deep Learning, and Statistical Models," *Environmental Science and Pollution Research*, vol. 32, pp. 2510–2535, 2025.
8. Y. Luo, "CO₂ Emission Prediction Based on Prophet, ARIMA and LSTM," in *Highlights in Science, Engineering and Technology (AMCCE 2023)*, vol. 76, p. 385, 2023.
9. T. Wen, Y. Liu, Y. Bai, and H. Liu, "Modeling and Forecasting CO₂ Emissions in China and Its Regions Using a Novel ARIMA-LSTM Model," *Heliyon*, vol. 9, no. 11, Art. e21241, 2023.
10. A. Ostermann, A. Bajrami, and A. Bogensperger, "Short-Term Forecasting of German Generation-Based CO₂ Emission Factors Using Parametric and Non-Parametric Time Series Models," *Energy Informatics*, vol. 7, Art. 2, 2024.
11. T. Li et al., "Reconstructing Global Daily CO₂ Emissions via Machine Learning," *arXiv*, 2024.
12. R. Jha, R. Jha, and M. Islam, "Forecasting US Data Center CO₂ Emissions Using AI Models: Emissions Reduction Strategies and Policy Recommendations," *Frontiers in Sustainability*, vol. 5, 2025.
13. Manaswini G. A., Poornima S. B., Bharathkumar G. B., Yuvaraj H. P., Bharth Kumar B. S., Hanumanthappa S. (2025). *Development of an intelligent mock interview platform using generative AI*. International Journal of Innovative Research in Technology, 12(7), 5227–5234. <https://doi.org/10.64643/IJIRT12I7-189276-459>
14. Yashaswini, N. B., Hanumanthappa, S., Ullas, M. P., Gunashree, H. K., Vismaya, K. R., & Rajashekar, K. J. (2025). *Abnormal behaviour detection in massive crowd*. International Journal of Innovative Research in Technology, 12(7), 1152–1158.