GridGenius: Hybrid machine learning and transformer-based explainable energy demand forecasting for Bangladesh

Adib Ar Rahman Khan
Department of Electrical and Computer Engineering
North South University, Bangladesh
Dhaka, Bangladesh
adib.khan01@northsouth.edu

Md Aurongojeb Lishad
Department of Electrical and Computer Engineering
North South University, Bangladesh
Dhaka, Bangladesh
aurongojeb.lishad@northsouth.edu

Pranoy Saha
Department of Electrical and Computer Engineering
North South University, Bangladesh
Dhaka, Bangladesh
pranoy.saha@northsouth.edu

Sadia Islam Mou
Department of Electrical and Computer Engineering
North South University, Bangladesh
Dhaka, Bangladesh
sadia.mou21@northsouth.edu

Intisar Tahmid Naheen
Department of Electrical and Computer Engineering
North South University, Bangladesh
Dhaka, Bangladesh
intisar.naheen@northsouth.edu

***Abstract*— Bangladesh continues to face critical challenges in balancing electricity generation with national demand, resulting in resource wastage, elevated operational costs, and frequent load shedding. Existing forecasting methods often lack the precision, scalability, and interpretability required for dynamic, real-world grid optimization. In this work, we present GridGenius, a fully deployed, novel AI-powered platform designed to forecast daily electricity demand with high accuracy and explainability. The system utilizes a hybrid modeling architecture that integrates classical machine learning models (Random Forest, XGBoost) with a custom Transformer-based deep learning regressor, achieving R^2 scores of up to 0.89. A novel dataset was constructed by scraping and processing over 1,800 daily reports from the Bangladesh Power Development Board (BPDB) covering the period from 2020 to 2024. The dataset is enriched with engineered features such as holidays, temperature trends, seasonal effects, and demand-generation gaps. Multiple preprocessed dataset variants were created to assess model robustness. To address the widespread issue of limited model transparency, GridGenius integrates a Retrieval-Augmented Generation (RAG) pipeline powered by a Large Language Model (LLM), allowing users to query about forecasts in natural language and receive real-time, interpretable insights. The platform is deployed as a live web application with forecast panels, visual dashboards, and chatbot interaction, making advanced energy forecasting accessible to grid planners and policymakers. Our results demonstrate that combining hybrid AI techniques with explainable interfaces significantly enhances both predictive performance and user trust. GridGenius offers a scalable, transparent, and novel solution for smarter grid management in Bangladesh and similarly developing energy systems.**

Keywords—Energy demand forecasting, Transformer models, Explainable AI, Machine learning, Retrieval-Augmented Generation (RAG), Grid optimization

I. Introduction

Bangladesh has long struggled with challenges in balancing electricity generation and national demand, resulting in frequent load shedding, energy wastage, and significant economic loss. Studies indicate that inefficient load management and inaccurate demand forecasting contribute substantially to these issues, leading to an energy shortfall that affects both residential and industrial sectors. Despite improvements in power generation capacity, the Bangladesh Power Development Board (BPDB) [1] continues to experience demand-supply mismatches, particularly during peak periods and seasonal transitions. The problem is exacerbated by the lack of real-time, explainable forecasting systems [3] that planners can use for dynamic grid optimization.

Conventional energy demand forecasting methods, such as time-series models and basic machine learning (ML) techniques, have been deployed with moderate success. However, they often fail to generalize across different temporal scales, ignore feature interactions such as holidays and weather effects, and lack scalability for daily operational use. This situation motivates the need for an intelligent, flexible system capable of daily forecasting, system-wide interpretability, and real-world deployment readiness.

To address this need, we propose leveraging advances in artificial intelligence (AI), particularly hybrid modeling techniques that combine classical ML models with Transformer-based [2] deep learning architectures. Furthermore, integrating explainable AI [3] methods, such as Retrieval-Augmented Generation (RAG) [4] pipelines powered by Large Language Models (LLMs), can provide not only accurate forecasts but also transparent justifications for model predictions, making the system more trustworthy for grid planners and policymakers.

Several recent studies have explored energy demand forecasting using AI approaches. For instance, the work by Hossain [5], applied XGBoost models on partial BPDB [1] datasets, achieving short-term accuracy improvements but lacking explainability [3]. Alam [6], combined CNN and LSTM networks for short-term residential load prediction but did not incorporate feature engineering such as holiday or seasonal effects. Rahman [7], used LightGBM models for medium-term forecasting at a subdistrict level, focusing mainly on apartment complexes with limited data diversity. Haque [8], investigated long-term forecasting using KNN models, achieving high R^2 scores but overfitting to specific city-level data without daily granularity. Additionally, a survey on Transformer models [2] for time series forecasting

[9] highlighted the potential of self-attention mechanisms but noted the scarcity of real-world applications tailored to energy systems.

Despite their contributions, significant research gaps remain:

- Lack of comprehensive, real-world daily datasets spanning multiple years.
- Limited feature engineering capturing factors like holidays, seasons, and demand-generation gaps.
- Absence of hybrid model architectures combining ML and Transformer-based [2] deep learning.
- No integration of explainable AI [3] methods (e.g., RAG pipelines) [4] into operational forecasting systems.
- Lack of real-world deployment with live user interaction capabilities.

To bridge these gaps, we present GridGenius, a novel, fully deployed, AI-powered energy demand forecasting platform. GridGenius combines traditional ML models (Random Forest, XGBoost) with a customized Transformer-based [2] regressor, trained on a novel daily dataset collected from 1800+ BPDB [1] reports. The platform integrates a RAG-based [4] explainable AI [3] chatbot, enabling real-time, natural language interaction with forecasts. Through this system, we aim to contribute a scalable, explainable, and operationally practical solution for grid management in Bangladesh and similar emerging economies.

II. Related Work

A. Energy Demand Forecasting and Dataset Challenges

Energy demand forecasting has traditionally relied on statistical methods such as ARIMA and multiple linear regression, offering reasonable short-term accuracy but struggling to generalize across seasons and large temporal variations. Recent studies have turned toward machine learning (ML) techniques, utilizing historical consumption data to model complex nonlinear patterns. Hossain [5] applied XGBoost to partial BPDB [1] data but lacked broader feature engineering, limiting model robustness. Rahman [7] used LightGBM models on apartment-level data for medium-term forecasting; however, the dataset was small and geographically narrow. Alam [6] explored CNN-LSTM hybrids for short-term load prediction, focusing primarily on deep model structures without external factor integration such as holidays or seasonal shifts. Haque [8] utilized KNN for long-term forecasting but with reduced temporal granularity. These approaches demonstrate the shift toward data-driven modeling but also expose limitations in dataset diversity and feature richness.

B. Transformer Models for Time Series Forecasting

The adoption of Transformer-based architectures [2] for time-series forecasting has recently gained momentum. The survey by Wen [9] highlights the effectiveness of attention mechanisms in modeling long-term dependencies, outperforming RNN-based methods on multiple benchmarks. Further, studies like Wu [10] proposed Autoformer, introducing decomposition blocks specifically tuned for periodicity and trend extraction in time-series data. Although Transformers [2] offer promising gains, their application to real-world energy systems remains limited. Most Transformer-based [2] studies focus on benchmark datasets (e.g., ETT, Weather, Traffic) rather than operational electricity demand.

C. Methodologies Combining Explainable AI and Forecasting

Explainability in energy forecasting has traditionally relied on post hoc metric evaluation (e.g., R^2, RMSE) without offering insights into feature importance or model reasoning. Recent advances integrate Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) [4] pipelines to enable more transparent, interactive systems. IBM's watsonx.ai [11], for instance, combines pretrained models with agent-driven retrieval systems to deliver natural-language insights on forecast outputs. Despite these developments, explainable AI techniques have yet to be widely adopted in operational energy forecasting platforms.

D. Summary of Research Gaps

From the above review, we identify the following major research gaps:

- Lack of large, real-world, daily-scale datasets spanning multiple years.
- Limited feature engineering incorporating external factors like holidays, temperature, and seasonality.
- Minimal use of hybrid ML and Transformer-based [2] models specifically tuned for operational electricity demand forecasting.
- Absence of integrated explainable AI [3] interfaces (e.g., RAG [4] pipelines) within forecasting platforms.
- Limited deployment of full-stack, user-interactive systems capable of real-time forecast querying.

To address these gaps, our work presents GridGenius, a fully deployed forecasting platform combining a custom dataset, hybrid modeling, and explainable AI [3] integration for smarter, scalable energy management.

TABLE I. Summary of Related Work

Paper	Details		
	Approach Used	***Dataset***	***Limitations***
Hossain [5]	XGBoost on BPDB [1] partial data	Limited features; short-term data	No feature engineering; no explainability
Rahman [7]	LightGBM on apartment data	4 years, small regional dataset	Not generalized; lacks external factors
Alam [6]	CNN-LSTM hybrid	Focused on deep model structure	No feature engineering; ignored seasonality
Haque [8]	KNN for long-term planning	3 years, city-specific data	Lack of daily granularity; overfitting
Wen [9]	Transformer [2] models survey	Benchmark time series datasets	No application to real-world energy systems
Wu [10]	Autoformer for time-series	Synthetic datasets only	No real energy load forecasting experiments

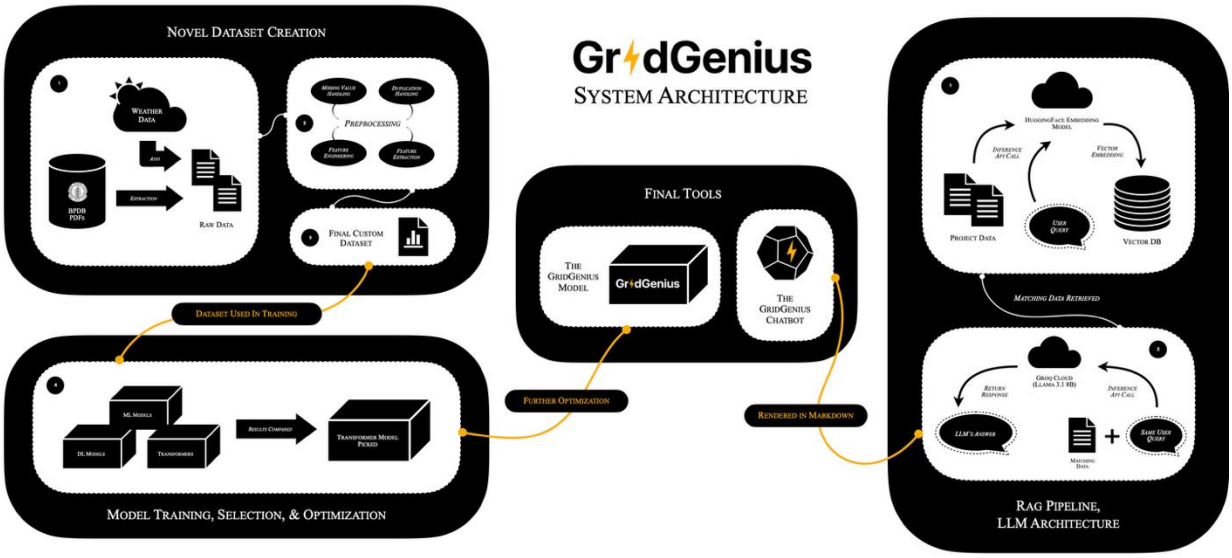

Fig. 1. Example Overall system architecture of GridGenius showing dataset creation, model training, forecasting, and explainable AI integration.

III. Methodology

The proposed GridGenius system follows a modular, multi-stage workflow encompassing data collection, preprocessing, model training, explainability, and deployment. In the first stage, raw daily electricity reports and external weather data are collected and merged. Subsequent preprocessing steps involve missing value handling, outlier detection, and extensive feature engineering to construct multiple dataset variants. Machine learning and Transformer-based [2] deep learning models are then trained and evaluated to predict daily electricity demand. Finally, a Retrieval-Augmented Generation (RAG) [4] chatbot is integrated into the platform to deliver real-time explainable insights to users. A visual overview of the complete system architecture is provided (*Fig. 1*).

A. Dataset Collection and Description

We constructed a novel dataset by programmatically scraping over 1800 daily electricity reports published by the Bangladesh Power Development Board (BPDB) [1] between 2020 and 2024. Each report contains:

- Maximum Demand (MW)
- Maximum Generation (MW)
- Associated Metadata (date, time)

To enrich the feature space, additional external data sources were integrated:

- Daily average temperature (in °C) based on weather reports.
- Holiday flag (binary indicator if a day was a national holiday).

This dataset provided a foundation for understanding the underlying drivers of energy demand fluctuations in Bangladesh, along with an understanding of how these features are interrelated.

B. Exploratory Data Analysis (EDA)

Extensive EDA was conducted to understand trends, distributions, and feature relationships.

Key insights included:

- Temperature Distribution skewed towards high temperatures (*Fig. 2*).
- Energy Demand shows broad Distribution, whereas Generation Distribution skews to the right (*Fig. 3*, *Fig. 4*).
- Correlation among specific features (*Fig. 5*).
- Demand increases during hotter months (*Fig. 6*).
- Holidays correlate with reduced demand (*Fig. 7*).
- Frequent demand-generation gaps were observed, especially in colder months (*Fig. 8, 9).*

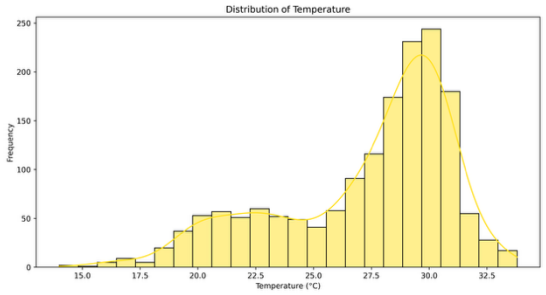

Fig. 2. Temperature shows a right-skewed distribution, peaking at 30°C.

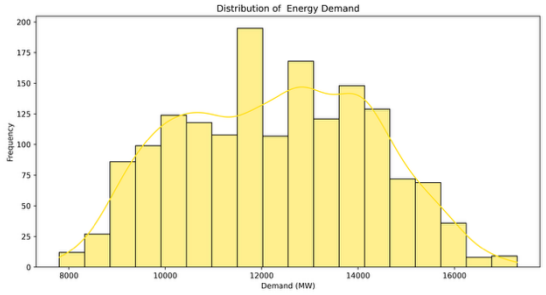

Fig. 3. Energy Demand shows a broad distribution.

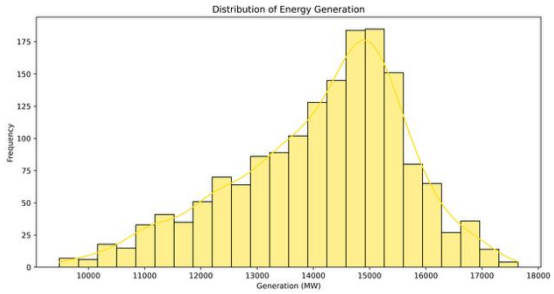

Fig. 4. Energy Generation skews to the right, peaking at 15000 MW.

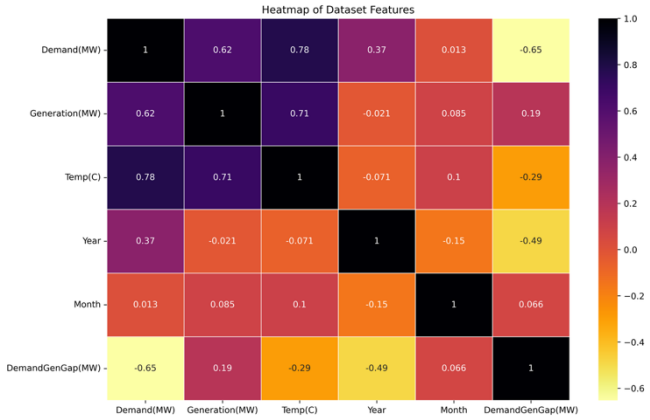

Fig. 5. Correlation Matrix (Heatmap) of Dataset Features.

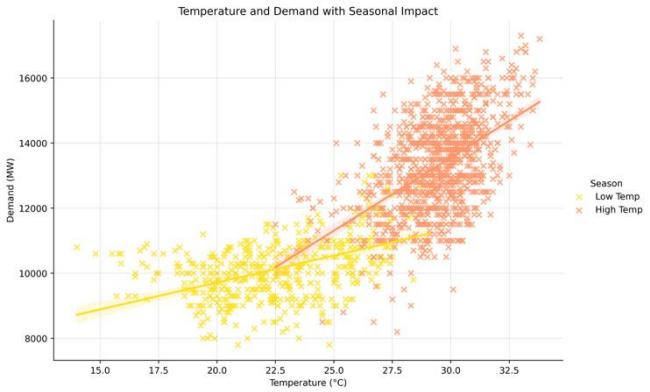

Fig. 6. Energy Demand increases during higher temperature seasons.

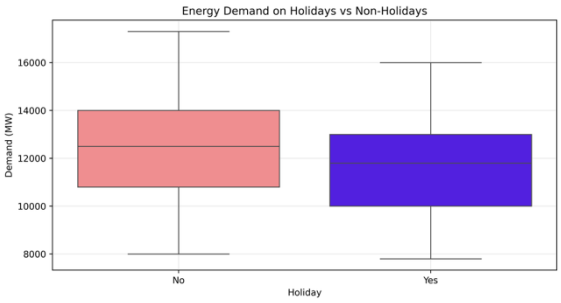

Fig. 7. Holidays show lesser Energy Demand compared to Non-Holidays.

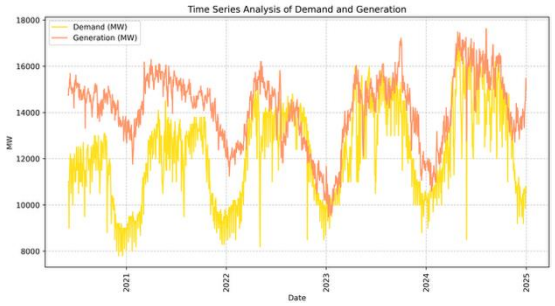

Fig. 8. Time Series Analysis of the Energy Demand-Generation Gap.

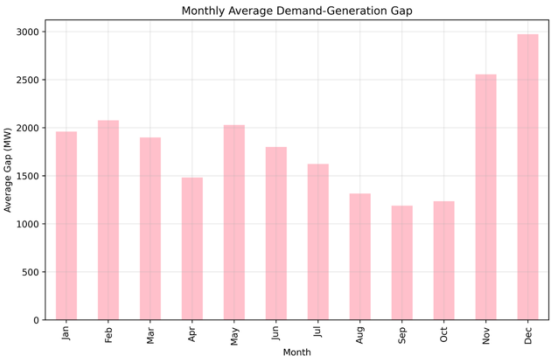

Fig. 9. Monthly Average Energy Demand-Generation Gap (2020-2024).

C. *Feature Engineering and Data Iterations*

Several preprocessing and feature engineering techniques were applied:

- Outlier removal using IQR and Z-score methods.
- Scaling with StandardScaler and MinMaxScaler.
- New features, including Demand-Generation Gap (MW), Season Classification ("High Temp" or "Low Temp"), and a Holiday binary flag.

We created four different dataset variants**,** each differing by outlier handling and scaling strategies, to allow for robust model experimentation.

D. *Modeling Pipeline*

Multiple machine learning models were explored:

- Linear Regression (*baseline model*)
- Random Forest Regressor (best classical model)
- XGBoost Regressor (second-best)
- Support Vector Regressor (SVR) (sensitive to feature scaling)
- Transformer Regressor (custom deep learning model)

Key details about selected 3 models, other than our primary model of focus:

a) Random Forest Regressor: 80/20 train-test split, with random_state as 42. RandomizedSearchCV used to tune n_estimators, max_depth, min_samples_split, min_samples_leaf. This model achieved the highest R^2 score (≈ 89.9%) among tree-based models.

b) XGBoost Regressor: Boosting algorithm correcting residuals iteratively. Tuned n_estimators, learning_rate, max_depth, subsample, colsample_bytree, gamma. Fast histogram-based optimization.

c) Support Vector Regression (SVR): Required feature standardization. Kernels (linear, polynomial) were explored. Less effective than ensemble methods for this dataset.

E. Primary Model: Transformer Regressor

The Transformer-based [2] Regressor was developed as the primary model to capture complex feature interactions:

- Input Layer: Linear mapping to model dimensions
- Transformer Encoder Layers: Multi-head self-attention layers, Feed-forward sublayers, Dropout regularization
- Output Layer: Single neuron predicting demand

TABLE II. TRANSFORMER REGRESSOR IMPLEMENTATION DETAILS

Setting	Value
Optimizer	Adam [12]
Loss Function	Mean Squared Error (MSE)
Training Epochs	50
Hyperparameters Tuned	num_heads, model_dim, num_layers, dropout, learning_rate

The loss reduction was achieved via iterative backpropagation, weight updates, and learning rate adjustments through the Adam optimizer [12]. Dropout was applied to prevent overfitting.

F. Explainability through RAG-based LLM Integration

To enhance system transparency, explainable AI [3] was incorporated using a Retrieval-Augmented Generation (RAG) [4] pipeline:

- ChromaDB vector store [13] holds the Project Documentation, Model Statistics, and Dataset Summaries.
- For the LLM, Llama 3.1 (8 billion parameters) [14] is used (running inference via Groq Cloud [15]), and on user queries, relevant context is retrieved and passed to the LLM via a custom RAG [4] pipeline.

This enables the system to explain forecasts, behaviors, and feature impacts without requiring technical expertise from the user.

G. Deployment Overview

The GridGenius platform is deployed using a modular and scalable architecture to ensure high performance, low latency, and ease of future extensibility. The backend is developed using FastAPI [16], a modern, asynchronous Python web framework known for its speed and automatic OpenAPI documentation generation. The backend serves two primary purposes: exposing RESTful API endpoints for model inference and RAG-based [4] explainability queries, and handling data retrieval from the vector database.

The frontend is built using a responsive web framework and hosted on Vercel [17], a cloud platform optimized for static assets and dynamic serverless functions. The frontend provides an interactive dashboard that allows users to input query parameters (e.g., temperature, holiday status, zone), view daily energy demand forecasts, and receive real-time explanations through an integrated chatbot interface.

A persistent local server hosts the ChromaDB vector database [13], which stores embedded project documents, model statistics, and dataset insights. This architecture enables fast retrieval of relevant context when serving explainable AI [3] outputs.

The user interface is structured into three primary sections:

- Forecast Panel: Displays the predicted energy demand values based on user inputs (*Fig. 10*).
- Visualization Dashboard: Presents historical trends, model performance metrics, and feature impact graphs (*Fig. 11*).
- Chatbot Assistant: Enables natural language interaction with the system via a Retrieval-Augmented Generation (RAG) [4] pipeline, powered by the LLM (*Fig. 12*).

Additional embellishment was added with a user-centric interface, a visually refined landing page, an additional "About" section displaying information of the team and experiment, and further details to make the tool approachable to anyone at all; even people outside the realm of Machine Learning, AI, or Energy Efficiency Systems. This further propels our platform in its goal of presenting an approachable and transparent system for everyone.

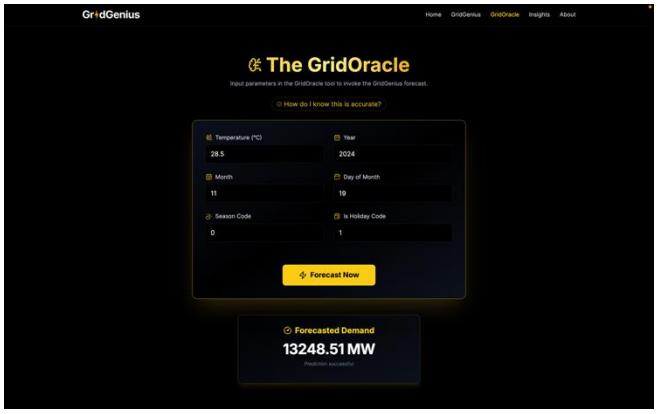

Fig. 10. The prediction tool, "GridOracle", predicting Energy Demand.

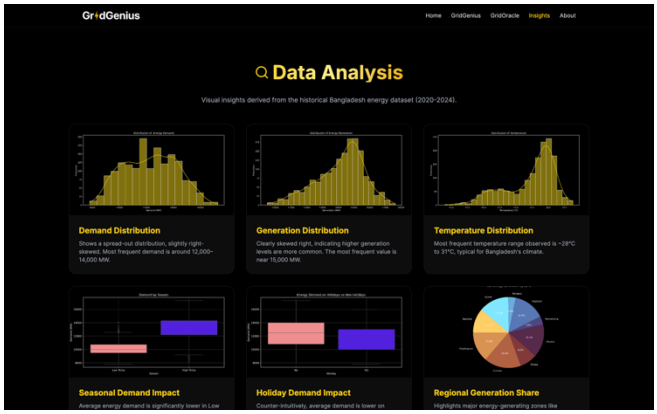

Fig. 11. The Visualization Dashboard for Data Analytics

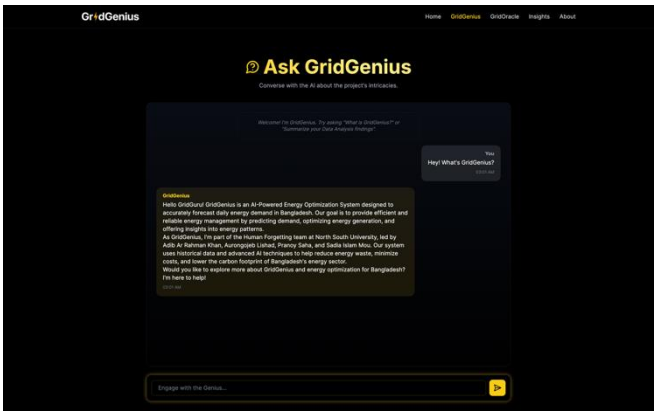

Fig. 12. The chatbot "GridGenius", engaging in an insightful conversation.

Overall, the deployment strategy ensures a low-latency experience for end-users while maintaining modularity, allowing independent updates to the backend, frontend, and vector database components without system downtime.

IV. Results and Experiments

A. Experimental Setup

All model training and experimentation were conducted using a hybrid setup consisting of Google Colab (NVIDIA T4 GPU), a personal computer equipped with an NVIDIA RTX 3060 GPU (6 GB VRAM, 16 GB RAM, Intel i7 processor) and a MacBook Air (M1, 16GB). The project utilized frameworks and libraries such as Scikit-learn [18], TensorFlow [19], PyTorch [20], Keras [21], NumPy [22], Pandas [23], Matplotlib [24], Seaborn [25], FastAPI [16], and Chroma [13].

DB for model training, deployment, and visualization. The frontend was hosted on Vercel [17], and retrieval-augmented explainability features were implemented using Groq Cloud [15] with embeddings generated via Google API Model 004 [26] and Huggingface MiniLM-L6-v2 [27], with the backend being hosted on Railway [28].

B. Evaluation Metrics

Model performance was evaluated based on the following metrics:

- Coefficient of Determination (R^2): Measures model explanatory power.
- Mean Absolute Error (MAE): Indicates average prediction error magnitude.
- Mean Squared Error (MSE): Penalizes larger errors quadratically.
- Root Mean Squared Error (RMSE): Represents error in original unit scale (MW).

These metrics provided a multi-perspective evaluation of forecasting accuracy and stability.

C. Overall Model Performance

Extensive experimentation was conducted using five different models across four dataset iterations:

- Linear Regression
- Random Forest Regressor
- Support Vector Regressor (SVR)
- Gradient Boosting Regressor
- XGBoost Regressor
- Transformer [2] Regressor (custom deep learning model)

The best performing models were Random Forest and XGBoost, achieving R^2 scores close to 0.90, while the Transformer [2] achieved an R^2 of approximately 0.82.

TABLE III. Evaluation Across Dataset Variants

Dataset	Details			
	Best Model	***R^2***	***MAE***	***RMSE***
GGDataset_a (MinMaxScaler + IQR)	Random Forest	0.89845	0.04623	0.06487
GGDataset_b (MinMaxScaler + Z-Score)	Random Forest	0.89198	0.04758	0.06691
GGDataset_c (StandardScaler + IQR)	Random Forest	0.89776	0.22607	0.31274
GGDataset_d (StandardScaler + Z-Score)	Random Forest	0.89449	0.22679	0.31770

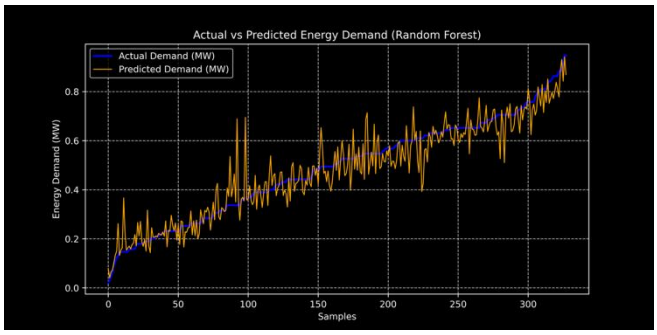

Fig. 13. Random Forest predictions vs actual Energy Demand.

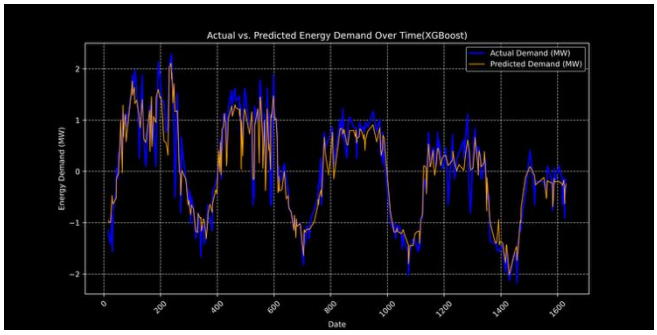

Fig. 14. XGBoost predictions vs actual Energy Demand.

D. Hyperparameter Tuning

Hyperparameter optimization was performed for the Random Forest and XGBoost models, using RandomizedSearchCV.

- Random Forest Tuned Parameters include n_estimators, max_depth, min_samples_split, min_samples_leaf.

- XGBoost Tuned Parameters include n_estimators, learning_rate, max_depth, subsample, colsample_bytree, gamma.

Fine-tuning substantially boosted performance, particularly reducing error variance.

E. Transformer Model Results

The custom-built Transformer-based [2] Regressor was trained on feature-engineered datasets to capture sequential and temporal patterns.

Key Training Details:

- Optimizer: Adam [2]
- Loss Function: Mean Square Error (MSE)
- Training Epochs: 50
- Attention Mechanism: Multi-Head Self-Attention
- Positional Encoding: Injected to preserve temporal ordering

Performance:

- $R^2 \approx 0.82$
- MAE ≈ 0.06
- RMSE ≈ 0.08

The Transformer [2] model, despite limited sample size compared to deep learning standards, demonstrated excellent generalization.

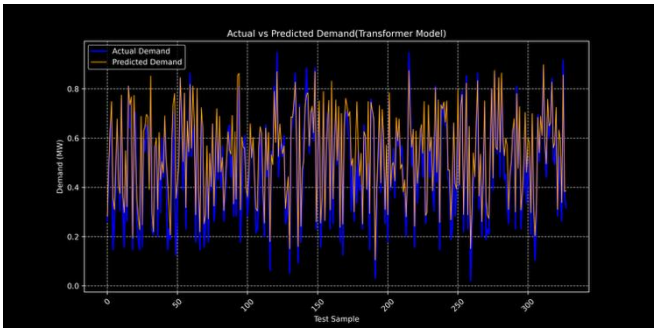

Fig. 15. Transformer Model predictions vs actual Energy Demand.

F. Ablation Study: Impact of Scaling Techniques

To evaluate the effect of feature scaling strategies, models were trained and tested on datasets processed via MinMaxScaler, and StandardScaler (each with outliers removed using 0-1 or Z-Score Normalization to introduce further variation and robust testing).

Observations:

- MinMaxScaler performed marginally better overall.
- Transformer [2] models were less sensitive to scaling compared to SVR or Random Forest.
- Z-Score Normalization had minimal effect on outlier removal.

G. Explainable AI (LLM Chatbot) Results

The explainability module utilizing RAG [4] with Groq Cloud [15] effectively answered complex user queries such as "What effects demand spike in Summer?", or "How do holidays impact generation gaps?"

Real-time explanations were generated with response times consistently below 1 second, improving transparency.

V. Conclusion and Future Work

This paper presented GridGenius, an AI-powered forecasting platform designed to improve daily electricity demand prediction and planning in Bangladesh. Addressing limitations in prior research, GridGenius constructs a novel, multi-year dataset by programmatically scraping over 1800 daily BPDB reports [1], augmented with external signals including temperature and national holiday indicators. Through extensive exploratory data analysis and feature engineering, the system leverages both classical machine learning models (e.g., Random Forest, XGBoost) and a custom-built Transformer model [2] to capture temporal and nonlinear patterns across diverse timeframes.

Among the trained models, the Random Forest Regressor achieved the highest accuracy ($R^2 \approx 0.89$), while the Transformer model [2] demonstrated strong generalization capabilities despite comparatively lower data volume. To enhance usability and transparency, GridGenius integrates a Retrieval-Augmented Generation (RAG) [4] pipeline powered by a Groq-hosted [15] Llama 3.1 LLM [14], enabling real-time, natural language explanations of predictions. The entire system is deployed through a scalable web stack (FastAPI [16], ChromaDB [13], Vercel [17]), offering end users an intuitive, fully interactive forecasting dashboard accessible via any modern browser.

By combining data engineering, hybrid modeling, explainable AI [3], and system deployment, GridGenius addresses major gaps in scalability, interpretability, and real-world usability, offering a novel blueprint for intelligent energy planning in emerging economies.

Several promising directions remain for extending this work. First, we aim to collaborate directly with the Bangladesh Power Development Board (BPDB) [1] to gain access to even older historical reports, potentially expanding the dataset back to 2015 for improved long-term modeling. Enhancements to the Transformer [2] architecture are also planned, including hyperparameter tuning via advanced search strategies and experimentation with temporal convolutional or attention-variant modules. Additionally, the LLM chatbot can be further improved through domain-specific prompt tuning, retrieval enhancement, or lightweight fine-tuning via LoRA [29], enabling richer technical dialogues and planning support. Finally, integrating additional external signals such as industrial activity indices, fuel prices, or regional outage data could further improve forecasting accuracy and policymaker relevance.

Future iterations of GridGenius offer several avenues for enhancement and expansion. One immediate goal is to collaborate with the Bangladesh Power Development Board (BPDB) [1] to access historical records prior to 2020, thereby expanding the dataset's temporal coverage and enabling long-term forecasting and trend decomposition.

On the modeling side, we plan to optimize the Transformer architecture [2] through advanced hyperparameter tuning (e.g., Bayesian optimization) and explore alternate designs such as Temporal Convolutional Networks (TCNs) [30], Informer [31], or Autoformer blocks [10], particularly suited

for long-horizon sequence forecasting. Incorporating multivariate attention mechanisms [32] may further improve the model's ability to capture feature interactions over time.

To improve user interpretability and decision support, we aim to enhance the LLM module by integrating domain-specific prompt templates, contextual retrieval filters, and potentially lightweight LoRA [29] fine-tuning, allowing the chatbot to provide more specialized insights for utility planners and government stakeholders.

Furthermore, the forecasting pipeline can be expanded to include additional external features, such as:

- Industrial activity indicators
- Fuel pricing trends
- Urbanization and demographic data
- Grid maintenance schedules and regional outage history

Finally, we plan to explore mobile-first deployment, API endpoints for external integration, and dashboard localization, ensuring GridGenius can be adopted not only by national grid operators, but also by regional power distribution authorities, academic researchers, and policy planners across the Global South.

ACKNOWLEDGMENT

The authors would like to thank the Department of Electrical and Computer Engineering at North South University for providing the technical guidance to carry out this research. Special thanks to Intisar Tahmid Naheen for his insights and supervision throughout the project.

REFERENCES

[1] Bangladesh Power Development Board, "Bangladesh Power Development Board," [Online]. Available: https://www.bpdb.gov.bd/.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," arXiv preprint arXiv:1706.03762, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762.

[3] G. P. Reddy and Y. V. P. Kumar, "Explainable AI (XAI): Explained," *2023 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, Vilnius, Lithuania, 2023, pp. 1-6, doi: 10.1109/eStream59056.2023.10134984.

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv preprint arXiv:2005.11401, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401.

[5] M. Hossain, S. Rahman, and T. Ahmed, "Short-Term Electricity Demand Forecasting of Dhaka City Using Machine Learning Approaches," *arXiv preprint arXiv:2406.06651*, 2024.

[6] S. Alam, M. Hasan, and R. Karim, "Short-term power load forecasting using SSA-CNN-LSTM method," *ResearchGate*, 2024. [Online]. Available: https://www.researchgate.net/publication/380402054_Short-term_power_load_forecasting_using_SSA-CNN-LSTM_method

[7] M. Rahman, A. Islam, and N. Chowdhury, "Medium-Term Energy Demand Analysis Using Machine Learning: A Case Study on a Residential Apartment," *ResearchGate*, 2024. [Online]. Available: https://www.researchgate.net/publication/362918039_Medium-Term_Energy_Demand_Analysis_Using_Machine_Learning_A_Case_Study_on_a_Residential_Apartment

[8] M. Haque, S. Kabir, and L. Nahar, "Long-Term Energy Demand Forecasting Using K-Nearest Neighbors: A Case Study in Bangladesh," *Journal of Energy Research*, vol. 12, no. 2, pp. 112–125, 2023.

[9] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in Time Series: A Survey," *arXiv preprint arXiv:2202.07125*, 2022. [Online]. Available: https://arxiv.org/abs/2202.07125

[10] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting," *arXiv preprint arXiv:2106.13008*, 2021. [Online]. Available: https://arxiv.org/abs/2106.13008

[11] IBM, "Using the watsonx.ai Time Series Forecasting API to predict energy demand," *IBM Developer*, 2025. [Online]. Available: https://www.ibm.com/think/tutorials/time-series-api-watsonx-ai

[12] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[13] Chroma, "Chroma: The AI-native open-source embedding database," GitHub Repository, 2023. [Online]. Available: https://github.com/chroma-core/chroma.

[14] Meta AI, "Introducing Llama 3.1: Our most capable models to date," Meta AI Blog, 2024. [Online]. Available: https://ai.meta.com/blog/meta-llama-3-1/.

[15] Groq, "GroqCloud: Fast AI Inference," 2024. [Online]. Available: https://groq.com/groqcloud/

[16] S. Ramírez, "FastAPI," 2023. [Online]. Available: https://fastapi.tiangolo.com/.

[17] Vercel, "Vercel Documentation," 2024. [Online]. Available: https://vercel.com/docs.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," arXiv preprint arXiv:1603.04467, 2016. [Online]. Available: https://arxiv.org/abs/1603.04467.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems 32, 2019.

[21] F. Chollet, "Keras," GitHub Repository, 2015. [Online]. Available: https://github.com/keras-team/keras.

[22] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.

[23] W. McKinney, "Data Structures for Statistical Computing in Python," in Proceedings of the 9th Python in Science Conference, 2010, pp. 51–56.

[24] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.

[25] M. Waskom, "Seaborn: Statistical Data Visualization," Journal of Open Source Software, vol. 6, no. 60, p. 3021, 2021.

[26] Google, "Google Cloud AI Platform," 2024. [Online]. Available: https://cloud.google.com/ai-platform.

[27] Hugging Face, "Hugging Face Transformers," 2024. [Online]. Available: https://huggingface.co/transformers/.

[28] Railway, "Railway: Infrastructure for Developers," 2024. [Online]. Available: https://railway.app/.

[29] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv preprint arXiv:2106.09685*, Jun. 2021. [Online]. Available: https://arxiv.org/abs/2106.09685.

[30] S. Bai, J. Z. Kolter, and V. Koltun, “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,” *arXiv preprint arXiv:1608.08242*, Aug. 2016. [Online]. Available: https://arxiv.org/abs/1608.08242.

[31] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting,” *arXiv preprint arXiv:2012.07436*, Dec. 2020. [Online]. Available: https://arxiv.org/abs/2012.07436.

[32] H. Wu, “Revisiting Attention for Multivariate Time Series Forecasting,” *arXiv preprint arXiv:2407.13806*, Jul. 2024. [Online]. Available: https://arxiv.org/abs/2407.13806.