A Data-Centric Approach to Multi-Region Urban Energy Demand Forecasting Using Ensemble Machine Learning

Dr. S Prince Chelladurai
Assistant Professor
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
princecs@srmist.edu.in

B S Vikash Srikanth
Student
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
vs7189@srmist.edu.in

S Ravi Prakash
Student
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
rp2382@srmist.edu.in

A Griffin George
Student
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
ga8488@srmist.edu.in

J Dharanidharan
Student
Dept. of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
dj7639@srmist.edu.in

***Abstract*—Abstract: With increased complexity in urban power grid systems, along with increased cost of errors in load forecasting, The vast majority current research focuses largely on the issue of picking the right model, paying little attention to the way the input data are prepared. In this study, we adopt a different approach. We believe that choosing a good input data preparation pipeline, in addition to choosing a good algorithm itself, is critical for obtaining a high-performing predictive system. We construct a pipeline of 23 features that includes temporal cyclical patterns, seasonality, historical load lags, and holiday indicators on five regional data sets from PJM Interconnection.Then, in order to ensure fairness and confidentiality during the assessment, we employ time-series aware cross-validation using GridSearchCV to evaluate five different XGBoost, and LightGBM etc. Even the simplest baseline model has an accuracy of 0.8517 while our top-performing algorithm Gradient Boosting of the PJME region achieves an R2 score of 0.8647 and MAPE of 5.61%.**

***Index Terms*—Energy Demand Forecasting, Feature Engineering, Ensemble Learning, XGBoost, LightGBM, GridSearchCV, Time Series, Data-Centric AI, PJM Interconnection.**

I. Introduction

ENERGYA demand forecast is the basis for electricity system operations in the contemporary world. The managers operating the electric grids all around the globe need to calculate every day the amount of electricity to generate in the upcoming hours, and every error in doing so might lead to very serious complications. Excessive electricity generation means wastage of fuel, as it is produced from an extra unit, whereas underestimation of electricity generation will result in brownouts/rolling blackouts. There is a cost involved in both situations, as the former leads to wastage of money while the latter leads to outage of power in towns/cities, among other issues. This has become even more difficult due to climate change and population increase.

II. Literature Review

The two communities of machine learning and power systems have been concentrating on the task of forecasting the energy demand for the last 20 years. Classical statistics have provided the base of initial attempts at computational modeling. The decade of 2000 saw wide use and even wider popularity of the Box-Jenkins family of algorithms based on the Auto-Regressive Integrated Moving Average approach, including its seasonal versions SARIMA. While not dealing well with the complex nature of interaction between weather, calendar events, and the behavior of people in generating real-life demand curves, these algorithms do great in case of a fairly static signal and near future predictions.

In another aspect of research, the use of feature engineering in the process was deemed to be vital to obtain high prediction accuracy, as evidenced by Hong and Fan [7].

III. System Architecture

The suggested design comprises of three different layers in order to keep the user interface, machine learning methods, and data management independent of each other.This will make it easy to optimize each individual layer separately. The pipeline method is then employed on the architectural structure. The feature engineering process receives the original hourly demand data from the CSV files and pre-processes them to develop a feature space consisting of 23 unique features. Once the pre-processing phase is completed, the processed datasets are sent to the model training module, where the GridSearchCV, together with TimeSeriesSplit, improves the five methods to avoid data leakage in time-series data. Six RESTful APIs using the FastAPI backend are offered to serialize and de-serialize the performance of the models. The output from the API is consumed by the client JavaScript-based dashboard on five distinct pages.

A. Data Pipeline Layer

The key of the system is in the data pipeline. Its simplicity, yet importance, can be understood through the necessity to clean and structure the dirty data from real life, represented in CSV files, and transform it into structured data that will allow training of the model. In total, five datasets, each representing a different region, go through this procedure. The first step of the procedure is data cleaning. Duplicates in timestamp values are found and deleted. It may sound strange, but duplicates are rather frequent in utility datasets because of incorrect logging or time zone differences. In order to fill possible gaps in data due to faulty sensors, up to three hours of data can be interpolated.

B. Machine Learning Layer

For each of the five processed regional data sets, the machine learning component is responsible for training, tuning, and evaluating five models. This yields a grand total of 25 trained model artifacts. The selected five models span a range of strategies and complexities:

- **Linear Regression**:the most basic approach that assumes a linear relation between features and demand. This defines the ceiling of performance achievable purely through the use of features without nonlinear modeling.
- **Random Forest**: a suite of decision trees based on bagging, which allows identifying nonlinear relations. They provide a natural way to rank feature importance and are immune to overfitting.
- **Gradient Boosting**:an iterative model-building process where each new tree corrects the errors of its predecessor. Though slower to build, it tends to outperform the Random Forest algorithm in terms of accuracy.
- **XGBoost**:A gradient boosting algorithm that successfully deals with sparsity and includes a mechanism of regularization. As far as practical machine learning is concerned, it has become an industry standard.
- **LightGBM**: For faster training, such a gradient boosting algorithm utilizes leaf-wise trees with histogram-based splitting. This makes it effective when working with large-scale data.

C. Frontend Layer

FastAPI acts as a backend providing the necessary static content for the front-end, which is represented by a one-page web application developed via simple HTML, CSS, and JavaScript. The charts are built with Chart.js, and asynchronous API requests are sent to the backend. The dashboard provides five distinct pages:

Demand Forecaster:the primary prediction interface. After selecting a specific region, model, date, and time, a prediction for the next 24 hours is produced taking into account a 48-hour historical context.

Model Comparison:predictions, error rates, and training performance in paired bar graphs and a ranked analysis table after applying all five models simultaneously on the same input data.

Data Explorer: provides data histograms, summary statistics, and past demand trends for different date spans (7, 30, 90, or 365 days) for each area.

Analytics: shows how demand is driven by identifying the structures behind demand behavior by looking at long-term hourly, daily, monthly, and yearly demand trends.

Saved Summaries: enables users to study and compare past forecasts by logging every forecast generated during a session through persistent local storage.

D. Backend Layer

The application uses FastAPI, a modern Python-based web framework with superior performance and automated API documentation capabilities, as the basis for the backend architecture. The web framework provides the static frontend assets, implements six RESTful APIs, and initializes and caches all 25 machine learning models when starting up. The six API endpoints are:

`GET /api/regions`: returns metadata, including date ranges, row counts, and average demand, for all regions accessible to the user.

`GET /api/models/{region}`: returns a list of all trained models and their performance statistics for a specified region.

`POST /api/predict`:returns predicted demand, actual demand, 48 hours of history, 24 hours of predictions, model metrics, and AI-generated story after processing the input region name, timestamp, and model name.

`POST /api/compare`: returns a ranking of all models that can be run for a specified region at a specified timestamp.

`GET /api/history/{region}`:historical demand data that have been sampled for EDA visualizations.

`GET /api/stats/{region}`: combined data concerning trends in demand in hourly, daily, monthly, and annual timeframes.

IV. Methodology

A. Dataset

The PJM Interconnection is a regional transmission organization responsible for the distribution of electricity across

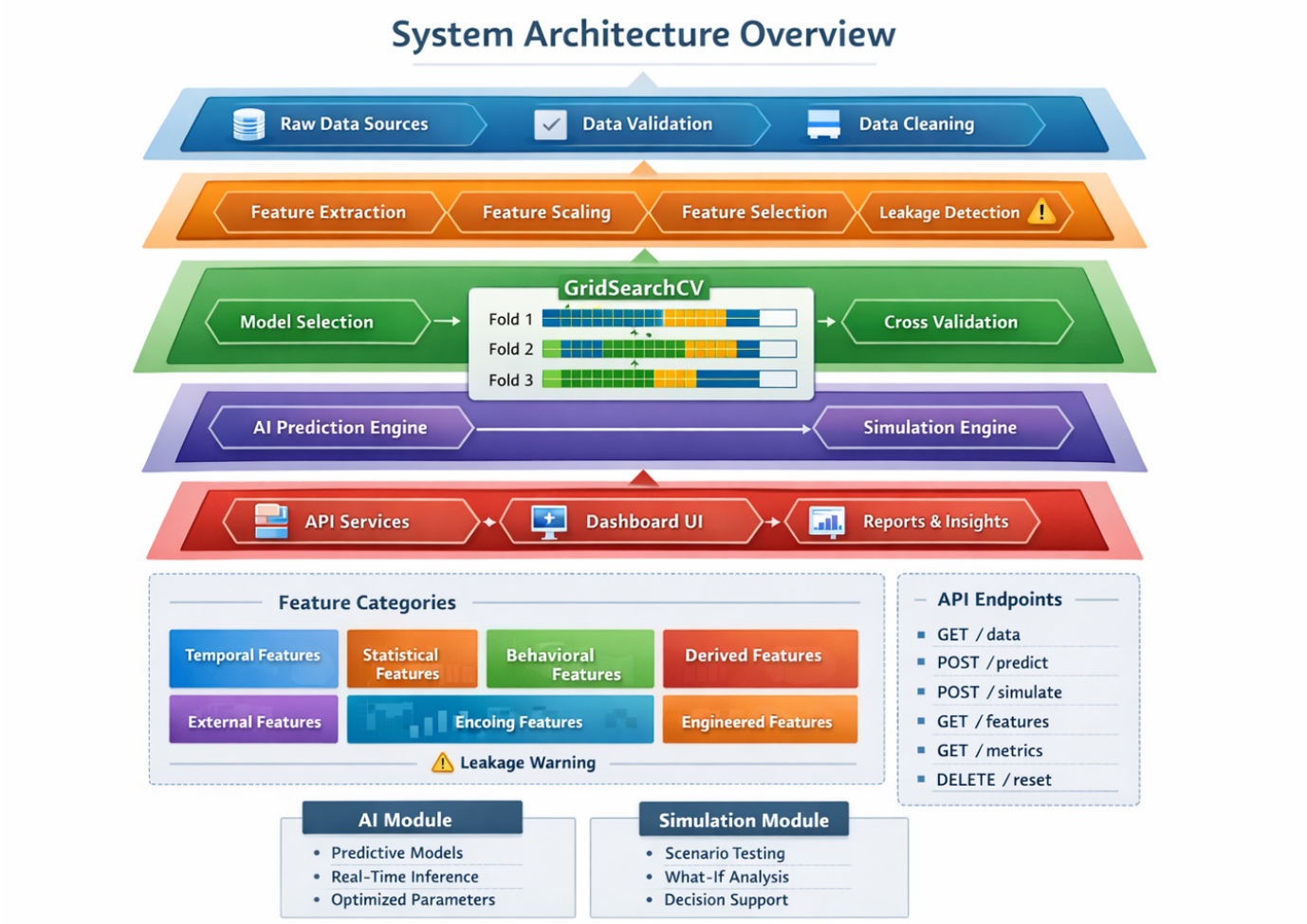

Fig. 1. End-to-end system architecture showing the data pipeline, multi-model training engine, and interactive dashboard.

13 different states in the east of the United States. The PJM Interconnection provides us with five-hourly demand statistics pertaining to electricity consumption. Due to the transparency, long period covered, and diversity of cases studied, PJM data are commonly used in energy forecasting literature. Both the

TABLE I
SUMMARY OF REGIONAL DATASETS

Region	Rows	Years	Avg MW	Peak MW
PJME	145,027	2002–2018	32,081	62,009
AEP	120,934	2004–2018	15,504	25,695
DAYTON	120,936	2004–2018	2,038	3,746
COMED	66,158	2011–2018	11,416	23,753
DOM	115,850	2005–2018	10,956	21,651

timestamps and the single demand value measured in megawatts per hour (MW) are included within the data sets. There is a wide variation with respect to the extent of the zones. While the zone DAYTON is small and made up of only a metropolitan area, having a demand of slightly over 2,000 MW, the PJME zone is relatively larger since it is composed of a region consisting of several states with demand of above 32,000 MW.

1) Data Cleaning: Before the training process, there are several quality issues that need to be sorted out regarding the raw data provided by PJM. The changeover period for daylight savings causes a shortage in hour counts during autumn and duplications during the spring season. In some of the data sets, demand is registered as "null" because of sporadic sensor failures. In addressing duplication issues, we retain the first occurrence while disregarding any others. We fill gaps that are up to three hours in length using forward fill and remove any remaining gaps.

2) Train-Test Split: An 80/20 time-based distribution approach is followed in all cases. While 20% of the dataset is always retained for testing purposes, 80% of the dataset is used for training purposes only. This is an ideal scenario in terms of the real world because it takes into account the forecasted demand through a historical database. To avoid any future leakage into the training dataset that would give very unrealistic results, we deliberately abstain from following a random distribution approach.

B. Feature Engineering

The next important part of the pipeline is the feature engineering part, which is what we believe to be our contribution in this task. While in most cases, the models are provided just raw data in the form of timestamps and demands, we generate 23 features, split into six different types, incorporating the domain knowledge related to the real life working of energy demand

TABLE II
COMPLETE FEATURE ENGINEERING TAXONOMY (23 FEATURES)

Category	Features	Count
Basic Temporal	hour, day_of_week, month, quarter, day_of_year, week_of_year	6
Cyclical Encoding	hour_sin, hour_cos, dow_sin, dow_cos, month_sin, month_cos	6
Binary Flags	is_weekend, is_peak, is_summer, is_winter, is_holiday	5
Historical Lags	lag_24h, lag_168h	2
Rolling Statistics	rolling_mean_24h, rolling_std_24h, rolling_mean_168h	3
Interaction	hour_x_weekend	1

1) Basic Temporal Features: Daytime is the foremost determinant of energy consumption. During early morning hours, like at three o'clock, when most individuals are sleeping, energy consumption levels are low. Energy consumption increases as the day progresses because factories and businesses open up. It reaches its peak level in the afternoon due to the high consumption levels of air conditioners. Energy consumption levels decline again during late hours of the day. Energy consumption varies on monthly basis (consumption levels are higher during summer months compared to spring and autumn months) and daily basis (consumption levels are high on weekdays and lower on weekends).

2) Cyclical Encoding: The issue here is that models consider timestamps as continuous data. Mathematically speaking, timestamps like 23 and 0 are not very far apart from each other temporally. It goes for weekends as Sunday and Monday and even months such as December and January. The issue is addressed by means of sinusoidal encoding:

$$x_{sin} = \sin\left(\frac{2\pi \cdot x}{T}\right), \quad x_{cos} = \cos\left(\frac{2\pi \cdot x}{T}\right) \tag{1}$$

where T is the time interval (24 if in hours, 7 if in days, and 12 if in months). No matter what numbers represent time intervals, adjacent time periods will always have comparable feature values because each temporal attribute gets mapped into a circle.

3) Binary Flags: Certain demand patterns are best captured as simple on/off indicators. We create five binary flags:

- **is_weekend**: 1 if today is either Saturday or Sunday. Otherwise, it is 0. Typically, demand is 8–12% lower on weekends than weekdays.
- **is_peak**: 1 if it is between 6 AM and 10 PM. It covers the extensive time period when the demand is high.
- **is_summer**: 1 if the current month is June, July, or August. Demand is higher due to the use of air conditioners during the summer.
- **is_winter**: 1 if it is December, January, or February. Electric heating results in an increase in demand during winter.
- **is_holiday**: 1 if today is a federal holiday in the United States. Usually, it results in a reduction in demand because of the shutdown of businesses, and the decrease is from 10

4) Historical Lags: The fact that current demand is influenced by the demand during the same period in the immediate past is expressed through lagged features. The following two lags are used:

- **lag_24h**: Demand was requested precisely 24 hours ago. This reflects the strong autocorrelation on the daily basis in the demand series.
- **lag_168h**:Demand was requested precisely 1 week ago (168 hours). This reflects the weekly pattern, where, for example, the same day in the week looks quite similar to its previous day in the week.

The conscious elimination of short-term delays became an important decision in our work. In our initial tests, we used delays of 1 hour and 2 hours, which led to almost perfect R^2 metrics of 0.9999. It was determined that these parameters were significantly distorting information about leakage: the algorithm memorized the previous value instead of identifying any relevant trends due to low fluctuations in demand. The R^2 metric fell from 0.9999 to around 0.86 after removing these parameters, yet now the models are real prediction algorithms that do not only reflect past values.

5) Rolling Statistics: We compute statistics for a moving window to allow the algorithm to get a feeling of recent demand trends without revealing the current-hour demand data:

- **rolling_mean_24h**:the mean demand level within the last 24 hours before the current time period.
- **rolling_std_24h**:Demand variation is reflected in its standard deviation during the same period.
- **rolling_mean_168h**: Context from a larger time period is gained by taking the average demand over the week's period.

This difference is important. Otherwise, the demand from the current hour will become part of the rolling window and data leakage will again occur.

6) Interaction Feature: There is an interaction feature, called `hour_x_weekend`. This is the multiplication of the weekend indicator with the hour variable. It is based on the discovery that the daily demand graph is different at the weekend compared to other days of the week. On weekday when workplaces are opened, there is a sharp rise early in the morning. At weekend, the level is low while the rise is more gradual.

C. Model Training and Evaluation

There will be 25 model artifacts created through training each of the five models using each of the five regional datasets.

To find out the optimal set of parameters for models which have parameters, we use GridSearchCV along with three folds of TimeSeriesSplit approach. The evaluation metrics used are:

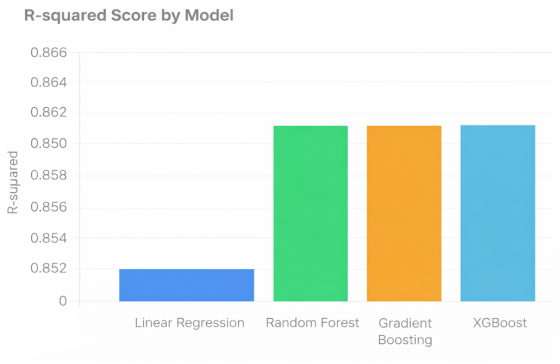

Fig. 2. R-squared scores across all five models for the PJME region, showing the performance hierarchy from Linear Regression to Gradient Boosting.

- **T**he ideal value of the score for a perfect model would be 1.0.
- **RMSE (Root Mean Squared Error)**: imposes a higher penalty on large forecast errors in terms of MWs.
- **MAE (Mean Absolute Error)**: In comparison to RMSE, the average forecast error in MW is more interpretable.
- **MAPE (Mean Absolute Percentage Error)**:It would come in handy when comparing accuracy percentages among regions having different magnitudes of demand values.

V. Implementation and Results

A. Training Configuration

Training and optimizing were done for five models sequentially for each of the five regions. It took 61.7 minutes on consumer grade computers for running the complete computation. The tree-based models that were implemented via GridSearchCV took between 3 to 10 minutes depending upon the size of the parameters grid and number of data samples. However, it only took less than one second for linear regression.

B. Results

Table III presents the complete performance comparison across all 25 model-region combinations. Several patterns emerge clearly. First of all, a good starting point is given by the feature engineering process. The R^2 values for all regions lie in the range from 0.80 to 0.85, despite using Linear Regression, which cannot capture nonlinear dependencies. Thus, one can see that the predictive value does not come from the machine learning algorithm learning complex decision boundaries but from the features created. Secondly, although the margins are tiny, typically ranging from 1 to 4 percentage points in R^2, the ensemble methods usually surpass linear regression. Hence, once the feature space is set properly, the advantages of model complexity diminish. Thirdly, in different regions,

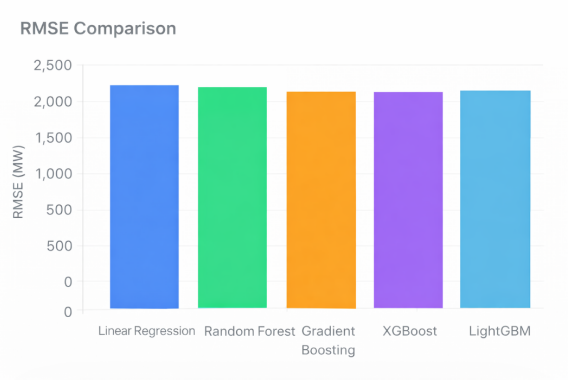

Fig. 3. RMSE comparison across all five models for each region, demonstrating consistent performance ranking.

either Gradient Boosting or XGBoost takes the leading place while LightGBM remains close behind. In any case, choosing any modern gradient boosting method is safe since they differ insignificantly most of the time.

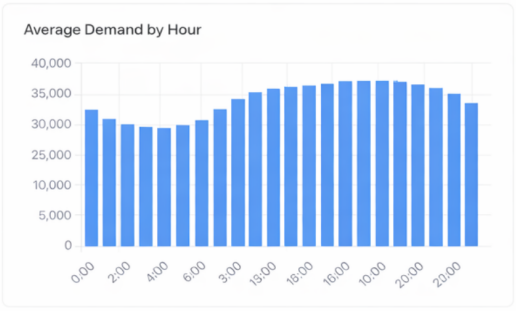

Fig. 4. Average hourly demand profile for PJME showing the characteristic daily consumption curve.

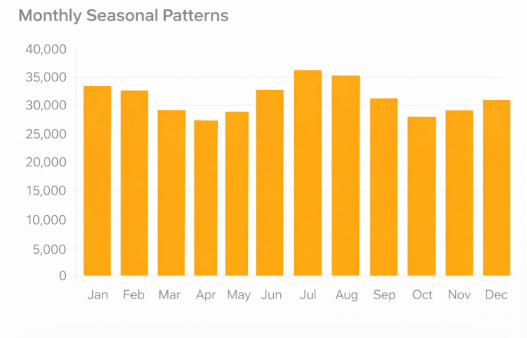

Fig. 5. Monthly seasonal demand variation across all five regions.

C. Data Leakage Analysis

Leakage in our data is an interesting phenomenon and one that we think deserves the attention of the scientific community. Our initial set of features included 1-hour and 2-hour lagged features (`lag_1h`, `lag_2h`), together with the 1-hour difference feature (`lag_1h_diff`). For all geographies, the models provided R^2 scores ranging from 0.9998 to 1.0000. This seemed great at first.

TABLE III
COMPLETE MODEL PERFORMANCE ACROSS ALL REGIONS

Region	Model	R^2	RMSE (MW)	MAE (MW)	MAPE (%)	Best Params
PJME	Linear Regression	0.8517	2,500	1,883	6.01	–
PJME	Random Forest	0.8610	2,420	1,768	5.59	depth=10, est=200
PJME	Gradient Boosting	**0.8647**	**2,388**	1,770	5.61	lr=0.05, depth=5, est=100
PJME	xgBoost	0.8644	2,390	1,771	5.62	lr=0.05, depth=5, est=100
PJME	Lightgbm	0.8645	2,390	1,771	5.62	lr=0.05, depth=5, est=100
AEP	Linear Regression	0.8303	1,008	789	5.40	–
AEP	Random Forest	0.8492	950	720	4.90	depth=10, est=200
AEP	Gradient Boosting	0.8512	944	727	4.98	lr=0.05, depth=5, est=100
AEP	xgBoost	**0.8524**	**940**	720	4.91	lr=0.05, depth=5, est=200
AEP	Lightgbm	0.8504	946	725	4.96	lr=0.05, depth=5, est=200
DAYTON	Linear Regression	0.8211	159	123	6.16	–
DAYTON	Random Forest	0.8565	142	106	5.27	depth=20, est=200
DAYTON	Gradient Boosting	0.8613	140	105	5.24	lr=0.05, depth=7, est=100
DAYTON	xgBoost	**0.8620**	**140**	105	5.23	lr=0.05, depth=7, est=100
DAYTON	Lightgbm	0.8601	141	106	5.31	lr=0.05, depth=-1, est=100
COMED	Linear Regression	0.8119	952	678	5.91	–
COMED	Random Forest	0.8188	935	612	5.18	depth=10, est=200
COMED	Gradient Boosting	0.8267	914	614	5.24	lr=0.05, depth=5, est=100
COMED	xgBoost	**0.8303**	**905**	608	5.19	lr=0.05, depth=5, est=100
COMED	Lightgbm	0.8290	908	610	5.21	lr=0.05, depth=5, est=100
DOM	Linear Regression	0.8018	1,102	815	7.15	–
DOM	Random Forest	0.8015	1,103	803	7.01	depth=10, est=200
DOM	Gradient Boosting	**0.8038**	**1,097**	806	7.05	lr=0.05, depth=3, est=200
DOM	xgBoost	0.8034	1,098	806	7.04	lr=0.1, depth=3, est=100
DOM	Lightgbm	0.8035	1,098	801	**6.98**	lr=0.05, depth=5, est=100

D. Anomaly Detection and Simulation

One interesting thing about our production model is its ability to simulate different anomalies. Anomaly simulation can be done in two ways, which are the following:

- **Holiday Simulation**: In the model simulation, the demand is estimated assuming the target day to be a public holiday by setting the variable `is_holiday` to 1. Because of the model having learned from previous data that holidays result in reduced demand, demand will generally reduce by 10 to 20 percent.
- **Grid Outage Simulation**:This technology creates the scenario of grid blackout by suddenly reducing the "expected" demand by 35%. The significant difference between expected and actual demand is detected by the AI Narrative Engine, after which the alert is issued by the technology that shows that this trend indicates infrastructure failure.

The benefits of this technology extend beyond just predicting; it can be used for training grid operators and testing the response of the forecasting process to anomalies through such simulations.

E. AI-Driven Demand Narrative

One of the most important parts of the AI-based predictive solution is the AI Demand Narrative engine. The backend

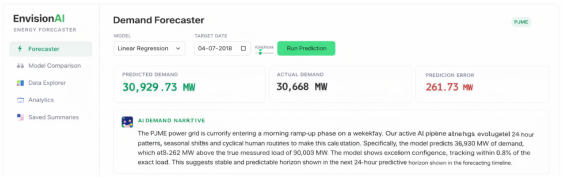

Fig. 6. Dashboard Forecaster interface showing prediction results with AI demand narrative.

Fig. 7. Model Comparison page showing side-by-side R^2, RMSE, and prediction accuracy for all five models.

generates a context-based narration that explains:

1) the current phase in the daily cycle of the grid (ramp-up, peak hours, ramp-down, etc.)
2) Operation environment (weekdays and weekends)
3) The differences that exist between the real figures and the forecast figures
4) Diagnosis of the size of the error and the causes of the error
5) Suggestions for monitoring the next 24-hour forecast period

In essence, this story links the output from the mathematical model to decision-making at the operational level, as the operator can easily digest the message without charts.

F. Advantages

Compared to the traditional method, the data-driven forecasting technique has the following significant benefits:

- **Honest and Transparent Evaluation**: The metrics provided here measure real forecasting ability and not artificially inflated values since we consciously remove any variables causing data leakage and adopt TimeSeriesSplit for cross-validation.
- **Multi-Region Generalizability**: The capability of our algorithm to generalize well is proved by the similar results obtained from the same 23-variable pipeline and model configurations in five distinct locations having highly varying power demands ranging from 2,000 MW to 32,000 MW.
- **Interpretable and Deployable**: Our algorithms based on decision trees automatically provide insights regarding feature importance without relying on advanced machine learning approaches. The whole framework is designed in a straightforward web application that runs on standard hardware.

G. Future Work

Several promising directions could extend this research:

1) **Weather Data Integration**: Adding features related to sun irradiation, temperature, and humidity on an hourly basis. This would improve R^2 by 5-10 percent just by itself, as per the relevant academic literature.
2) **Deep Learning Comparison**:In order to determine whether architecture complexity offers any advantage over feature engineering, LSTM and Transformer-based models are developed using the same 23-variable pipeline.

VI. Conclusion

In order to forecast the energy consumption of cities, this study used an approach that emphasizes feature engineering over model complexity. The ability to obtain decent forecasting performance was demonstrated through the development of a feature set consisting of 23 features reflecting the dynamics of power consumption. Gradient Boosting and XGBoost deliver the best performance, scoring an average R^2 between 0.80–0.86 and MAPE of 5-7% without weather information for any of the five PJM region datasets analyzed in five machine learning techniques.

References

[1] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ: Wiley, 2015.

[2] H. Hippert, C. Pedreira, and R. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 44–55, 2001.

[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 785–794, 2016.

[4] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 3146–3154, 2017.

[5] W. Kong et al., "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[6] sieh-Fu Tsai, Evaluation of EGFR and RTK Signaling in the Electrotaxis of Lung Adenocarcinoma Cells under Direct-Current Electric Field Stimulation,PLOS one.2022

[7] Explainable AI for Healthcare 5.0: Opportunities and Challenges Deepti Saraswat; Pronaya Bhattacharya, IEEE access 2022

[8] Explainable AI in prediction of electricity,Urja Pawar; Donna O'Shea, IEEE Explore 202

[9] From Blackbox to Explainable AI in Healthcare: Existing Tools and CaseStudiesParvathaneniNaga Srinivasu , 1N. Sandhya, august 2023

[10] Deep Learning Innovations in Detection of Lung Cancer: Advances, Trends and Open Challenges, Cognitive Computation, https://dataminer2.pjm.com/. [Accessed: Apr. 10, 2026].