NORTH SOUTH UNIVERSITY

Center of Excellence in Higher Education

📘 Project Report on

Zone-wise Electricity Demand Prediction using Machine Learning

Course Code: CSE 445
Course Title: Machine Learning
Section: 01
Course Instructor: Dr. Sifat Momen

Submitted by: Hasibul Hasan

Submission Date: April 20, 2025

Group Number: 6

Group Members:

Name	ID
Hasibul Hasan	**2021152642**
Al Imran	**2212602642**
Momtahina Quyyum	**2211889042**
Shadman Shakib Dip	**2111882642**

Department of Electrical and Computer Engineering
North South University

1. Abstract

The growing demand for electricity in Bangladesh has placed immense pressure on the national grid, particularly in densely populated and industrial zones. Efficient energy distribution remains a major challenge for the Bangladesh Power Development Board (BPDB), often leading to power shortage and unplanned load shedding [1]. In this project, we aim to develop a machine learning-based solution to predict electricity demand on a zone-wise basis, using historical data collected from BPDB records spanning March 2021 to March 2025 [2].

Our approach involves analyzing key features such as time-based consumption patterns, zone classifications, weather conditions, and supply shortages. Through data preprocessing, Exploratory Data Analysis (EDA), and model training, we evaluate multiple supervised learning algorithms, including Linear Regression, Random Forest, and XGBoost to identify the most accurate model for forecasting regional demand. The goal is to help optimize power distribution, reduce imbalances, and assist grid operators in making more informed decisions.

By providing actionable insights and accurate forecasts, our project contributes toward smarter electricity management and supports BPDB's ongoing efforts to improve power system reliability across Bangladesh.

2. Introduction

Electricity is one of the most essential needs in our daily lives, and its proper management is critical for the development of any country. In Bangladesh, the responsibility of generating and distributing electricity lies mainly with the **Bangladesh Power Development Board (BPDB).** As the demand for electricity continues to grow, especially in urban and industrial zones, the challenges of managing supply and demand becomes more complex.

BPDB provides a **daily forecast of total electricity demand** on their official website [3]. However, this prediction is not divided by zones or regions, and in most cases, it **does not match the actual demand** on the ground. This mismatch often leads to supply shortages, inefficient distribution, and unexpected load shedding in different parts of the country.

To solve this problem, we have taken a data-driven approach using **machine learning** to predict **zone-wise electricity demand** more accurately. By analyzing historical electricity data, weather conditions, and supply shortages from March 2021 to March 2025, our goal is to build a reliable model that can forecast the demand in each zone separately. This kind of prediction can help authorities distribute electricity more effectively, reduce wastage, and ensure better service to the people of Bangladesh [4].

This report explains the steps we took to develop this model, the challenges we faced, and the results we achieved using different machine learning algorithms.

3. Literature Review

Accurate electricity demand forecasting is crucial for efficient energy planning, load balancing, and minimizing wastage in the power system. This is particularly important in average-growing economies like Bangladesh, where rapid urbanization, population growth, and industrial expansion are continuously reshaping electricity consumption patterns. Traditional statistical methods such as **ARIMA** or manual estimation often fall short in capturing nonlinear consumption trends, regional disparities, and external variables like temperature or rainfall. This has led researchers to adopt **machine learning (ML)** techniques, which can process complex relationships and uncover patterns that are otherwise difficult to model manually.

In the study “Forecasting Energy Demand with machine Learning” by Joaquín Amat Rodrigo and Javier Escobar Ortiz, the authors provided a detailed comparison between classical models (like ARIMA and SARIMAX) and machine learning algorithms such as XGBoost, LightGBM, and deep learning architectures like LSTM.They emphasized the importance of choosing the right forecasting strategy, recursive, direct multi-step, or multi-output based on the problem type and data structure. Their results showed that tree-based and deep learning models consistently outperformed traditional approaches when dealing with time series electricity data, especially for long-term or multi-step forecasts [5].

Bringing the focus to Bangladesh, the 2023 IEEE conference paper titled “Energy Demand Forecasting Using Machine Learning: Perspective Bangladesh” by Avijit Paul Piya and colleagues tackled the problem using real national load data from BPDB over a span of 11 years. They experimented with LSTM, SARIMAX, and Facebook Prophet, and found that LSTM offered the best results in terms of RMSE and MAPE. Notably, their study highlighted a gap in the BPDB’s official forecasting model, while the board provides daily national forecasts, it lacks detailed zone-wise predictions, which are crucial for managing supply at the local level [6].

Another strong contribution came from the work titled “Electricity Consumption Prediction Using Machine Learning” by Vijendar Reddy et al., where multiple ML model, Linear Regression, K-Nearest neighbors (KNN), Random Forest, XGBoost, and ANN, where applied to hourly electricity usage data. Among all, KNN delivered the highest prediction accuracy of 90.92%. The paper also stressed the importance of robust preprocessing (handling outliers and missing values) and using metrics like RMSE and R^2 to evaluate model performance comprehensive [7]. This study shows that even simpler ML models can produce powerful results when combined with proper data handling techniques.

In addition to academic research, an undergraduate capstone project from Brac University made notable strides toward a practical solution. Their paper, “Modelling and Forecasting Energy Demand of Bangladesh Using AI-Based Algorithms,” introduced a tool that automatically scrapes and processes load data from BPDB’s daily PDF reports. This was particularly innovative given the lack of publicly available, structured historical datasets. Using this dataset, they tested KNN, Random Forest, and LSTM models, with LSTM again emerging as the most accurate. Their work is especially commendable for combining data engineering, machine learning, and real-world application, all tailored to the Bangladeshi context [8].

Despite these valuable efforts, most studies still focus only on aggregate national-level forecasts. However, power consumption behavior can differ significantly from one zone to another due to climate, industry presence, and population density. Currently, BPDB does not provide zone-wise forecasting publicly, creating inefficiencies in power allocation and frequently mismatches between supply and demand in local grids.

This gap inspired our project's direction. Building upon these foundational studies, we aim to apply machine learning models, especially Linear Regression, XGBoost, Random Forest on zone-level BPDB data to predict electricity demand with greater accuracy. By integrating additional variables like weather data and real-time supply shortages, our system hopes to empower regional grid operators with actionable insights for smarter, more localized energy distribution.

4. Methodology

This section describes the step-by-step approach we followed to build a machine learning model for zone-wise electricity demand prediction. It includes details about how we collected and prepared the data, explored key patterns and trends, selected appropriate algorithms, trained and evaluated the models, and interpreted the results. Each stage was carefully designed to ensure the accuracy and reliability of our predictions.

4.1 Data collection

The dataset for our project was collected from the official website of the **Bangladesh Power Development Board (BPDB)**. We manually extracted daily electricity data ranging from **March 2021 to March 2025**, which included valuable information related to electricity generation, demand, supply, weather, and fuel shortage across different zones in Bangladesh (look at **Figure 4.2**).

A sample of the data source as displayed on the BPDB website is shown in **Figure 4.1**. The data was mostly available in PDF format and required manual extraction and cleaning.

(C) Actual data of 11.03.25 Yesterday Tuesday :

01.	Max. Demand at eve. peak (Generation end)	:	13284 MW, at =	21:00 Hr.
02.	Evening-peak Generation (Generation end)	:	13284 MW, at =	21:00 Hr.
03.	Highest Generation (Generation end)	:	13284 MW, at =	21:00 Hr.
04.	Day Peak Demand	:	12144 MW, at =	12:00 Hr.
05.	Day-peak Generation (Generation end)	:	12101 MW, at =	12:00 Hr.
06.	Minimum Generation (Generation end)	:	9719 MW, at =	06:00 Hr.
07.	Generation shortfall at evening peak due to :	:		
	a) Gas/LF limitation	:	3857 MW	
	d) Coal supply Limitation	:	614 MW	
	b) Low water level in Kaptai lake	:	190 MW	
	c) Plants under shut down/ maintenance	:	2764 MW	
08.	Total Energy (Generation + Import)	:	288.86 MKWh	
	By Gas = 125.658 MKWH		By Oil = 26.490 MKWh	
	By Coal = 80.095 MKWH		Hydro&Wind= 1.247 MKWh	
	By Solar= 3.826 MKWH		Imported= 51.547 MKWh	
09.	Total Gas Supplied	:	937 MMCFD	

10. Zone wise Demand and Load-shed at Evening Peak (Sub-station end) :

Zone	Demand MW	Supply MW	Load Shed MW	Zone	Demand MW	Supply MW	Load Shed MW
Dhaka	4879	4879	0	Mymensingh	1044	1044	0
Chattogram	1318	1318	0	Sylhet	446	446	0
Khulna	1467	1467	0	Barishal	366	366	0
Rajshahi	1298	1298	0	Rangpur	721	721	0
Cumilla	1200	1200	0				
				Total	12739	12739	0

11.	Fuel cost :	(a) Gas =	447756000 Taka	(c) Coal =	573481155	Taka
		(b) Oil =	469613133 Taka	(d) Renewable	65892134	Taka
		(e)Import=	292020091 Taka	Total =	1848762512	Taka
12.	Maximum Temperature:		34.3			
13.	Energy Flow from East to West:		0			
14.	Energy Flow from West to East:		6661091.008			
15.	Power Flow during Peak Demand:		572			
16.	Maximum Power Flow:		926			

(D) Forecast of 12-03-2025 (Today) Wednesda :

01.	Probable Maximum Demand at Day Peak:	12380.00 MW	04. Probable Load Shed:	0 At evening peak (Sub-station end)
02.	Probable Maximum Demand at Evening Peak:	13230.00 MW	05. Probable Total Energy Generation:	284.99 MKWHr.
03.	Probable Maximum Generation at Evening Peak:	13230.00 MW	06. Probable Maximum Temperature:	34.00°C

* Captive Power ** Imported Power

#Remarks: Highest Generation 16477 MW on 30-04-2024 at 21:00

(Md. Helalur Rahman)
Deputy Secretary, Generation

Figure 4.1: Screenshot of daily electricity data from the BPDB website

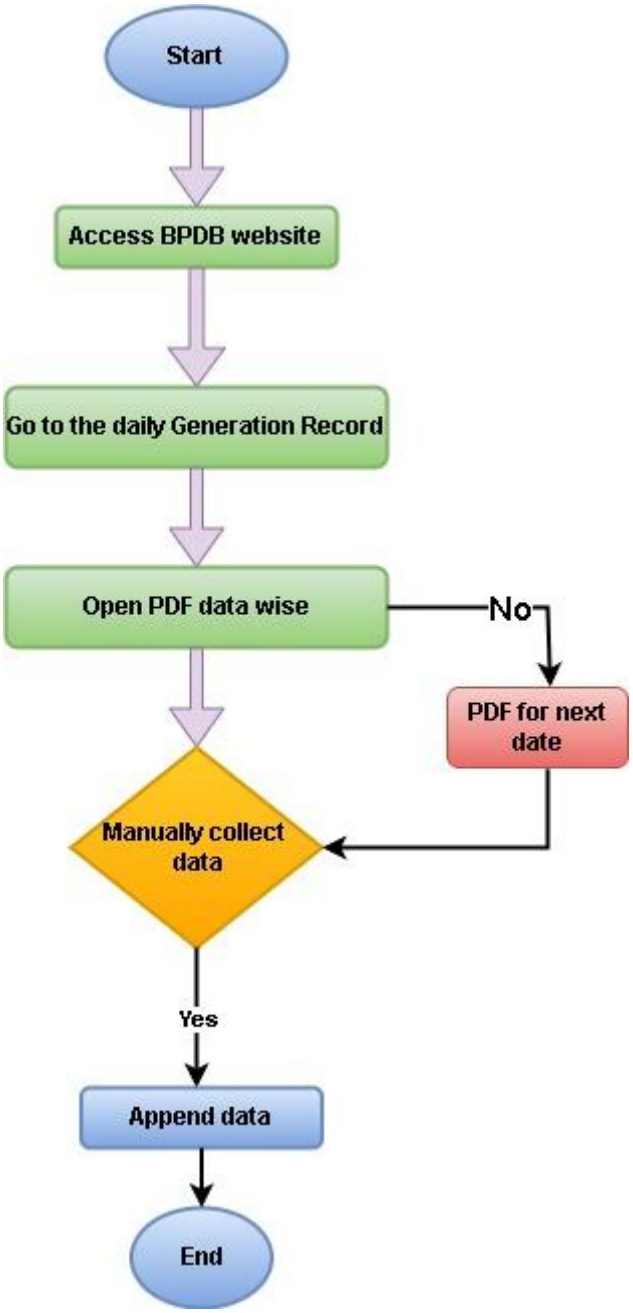
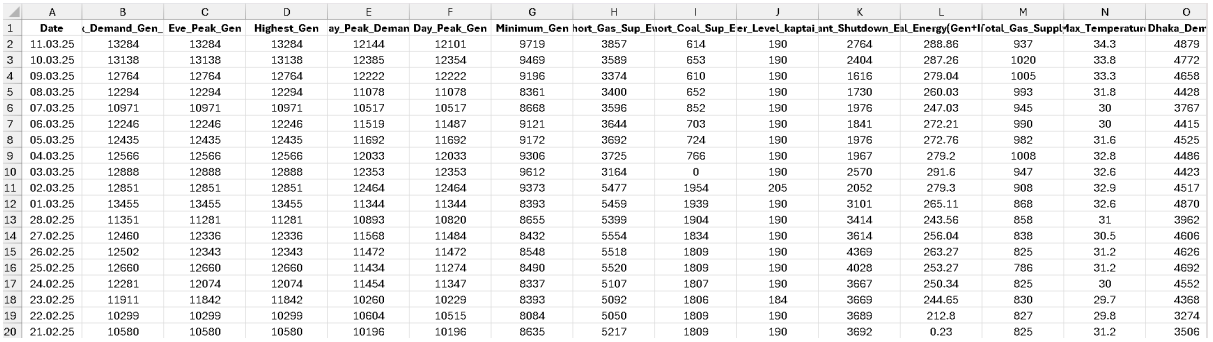


Figure 4.2: Data extraction process

After extraction, we compiled the data into a structured format, as shown in **Figure 4.3**. This dataset includes daily records with zone-wise demand and supply values, along with other relevant features such as fuel shortages, temperature, and generation statistics.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Date	_Demand_Gen_	Eve_Peak_Gen	Highest_Gen	ay_Peak_Deman	Day_Peak_Gen	Minimum_Gen	hort_Gas_Sup_E	ort_Coal_Sup_E	er_Level_kaptai	ant_Shutdown_E	al_Energy(Gen+I	Total_Gas_Suppl	Max_Temperatur	Dhaka_Dem
2	11.03.25	13284	13284	13284	12144	12101	9719	3857	614	190	2764	288.86	937	34.3	4879
3	10.03.25	13138	13138	13138	12385	12354	9469	3589	653	190	2404	287.26	1020	33.8	4772
4	09.03.25	12764	12764	12764	12222	12222	9196	3374	610	190	1616	279.04	1005	33.3	4658
5	08.03.25	12294	12294	12294	11078	11078	8361	3400	652	190	1730	260.03	993	31.8	4428
6	07.03.25	10971	10971	10971	10517	10517	8668	3596	852	190	1976	247.03	945	30	3767
7	06.03.25	12246	12246	12246	11519	11487	9121	3644	703	190	1841	272.21	990	30	4415
8	05.03.25	12435	12435	12435	11692	11692	9172	3692	724	190	1976	272.76	982	31.6	4525
9	04.03.25	12566	12566	12566	12033	12033	9306	3725	766	190	1967	279.2	1008	32.8	4486
10	03.03.25	12888	12888	12888	12353	12353	9612	3164	0	190	2570	291.6	947	32.6	4423
11	02.03.25	12851	12851	12851	12464	12464	9373	5477	1954	205	2052	279.3	908	32.9	4517
12	01.03.25	13455	13455	13455	11344	11344	8393	5459	1939	190	3101	265.11	868	32.6	4870
13	28.02.25	11351	11281	11281	10893	10820	8655	5399	1904	190	3414	243.56	858	31	3962
14	27.02.25	12460	12336	12336	11568	11484	8432	5554	1834	190	3614	256.04	838	30.5	4606
15	26.02.25	12502	12343	12343	11472	11472	8548	5518	1809	190	4369	263.27	825	31.2	4626
16	25.02.25	12660	12660	12660	11434	11274	8490	5520	1809	190	4028	253.27	786	31.2	4692
17	24.02.25	12281	12074	12074	11454	11347	8337	5107	1807	190	3667	250.34	825	30	4552
18	23.02.25	11911	11842	11842	10260	10229	8393	5092	1806	184	3669	244.65	830	29.7	4368
19	22.02.25	10299	10299	10299	10604	10515	8084	5050	1809	190	3689	212.8	827	29.8	3274
20	21.02.25	10580	10580	10580	10196	10196	8635	5217	1809	190	3692	0.23	825	31.2	3506

Figure 4.3: Preview of the structured dataset after extraction and organization

The dataset was rich in features and structured with columns such as:

- **Date-wise generation metrics:** Maximum demand generation in evening, Highest generation, Minimum generation, Day peak generation, Evening peak generation
- **Fuel-wise resource conditions:** Shortage of gas supply and coal supply, water level at Kaptai, plant shutdown and total gas supply.
- **Zone-wise demand, supply, and load shedding:** Dhaka demand, supply and load shedding at evening (similar for Chattogram, Khulna, Rajshahi, Cumilla, Mymensingh, Sylhet, Barisal and Rangpur).
- **Weather:** Maximum temperature
- **National totals:** Total demand, supply and load shedding in the evening.

4.2 Data Preprocessing

After collecting the raw data, we performed a series of preprocessing steps to clean and prepare it for further analysis and machine learning tasks. The main goals were to handle missing or inconsistent values, convert data types appropriately, and structure the dataset for time-series analysis.

Date Conversion and indexing: The **Date** column was originally in string format **(dd.mm.yy)**. To enable time-based analysis, we converted it into **datetime** format using **pandas**. We also set the Date column as the index of the DataFrame, which made it easier to work with time series data and perform operations like filtering by day, month, weekday and weekend.

Handling Missing and Invalid Values: Some values in the dataset were marked as **"N/A"** due to missing records in the source PDFs. We first replaced all such entries with **NaN**, and then filled the missing values using the **median** of each column. The mathematical formula used to calculate the median during imputation.

$$\text{Median} = \begin{cases} X[\frac{n+1}{2}] & \text{if n is odd} \\ \frac{X[\frac{n}{2}]+X[\frac{n}{2}+1]}{2} & \text{if n is even} \end{cases}$$

where

- **n** is number of observations in a data set
- **X** is the ordered/sorted list of values in the data set

The median is calculated by sorting the values and taking the middle one (or the average of two middle values if the count is even), making it a reliable way to fill missing data without being affected by outliers.

After the above steps, we checked for any remaining missing values to ensure the dataset was fully cleaned. We also verified the shape of the final dataset to confirm that no essential data was accidentally dropped.

These preprocessing steps helped us build a clean, consistent, and time-aware dataset, which was essential for generating accurate visualizations and training our machine learning models effectively.

4.3 Exploratory Data Analysis (EDA)

To understand the structure, patterns, and relationships within our dataset, we performed exploratory data analysis using the **ydata-profiling** library. This tool automatically generated a detailed HTML report that provided key statistics, visualizations, and data quality insights.

From the profiling report, we identify several important observations:

The dataset contains **42 numeric variables** and **1461 daily records**, covering a four-year period from March 2021 to March 2025. Notably, there were **no missing values**, and only **0.5% duplicate rows**, which indicates the dataset was mostly clean and consistent after preprocessing. **(see Fig. 4.4)**

Dataset statistics

Number of variables	42
Number of observations	1461
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	8
Duplicate rows (%)	0.5%
Total size in memory	490.8 KiB
Average record size in memory	344.0 B

Fig 4.4: Dataset summary showing variable types, record count, duplicates, and memory usage

Here are some key insights from the profiling report:

- **High Correlation Between Core Features:** Many features showed strong correlations with each other, particularly between total and zone-wise electricity **demand**, **supply**, and **load shedding**. This confirms that demand and supply generally moved together and helped us focus on the most impactful variables for modeling.
- **Max Demand, Highest Generation, and Minimum Generation with time series:** To better understand the trend in the electricity demand and generation over time, we visualize maximum demand, highest generation, and minimum generation values over the four-years period. As shown in **Figure 4.5-4.7**, electricity consumption and production exhibit clear seasonal and annual fluctuations, with peak typically occurring during summer months due to increased cooling demands.

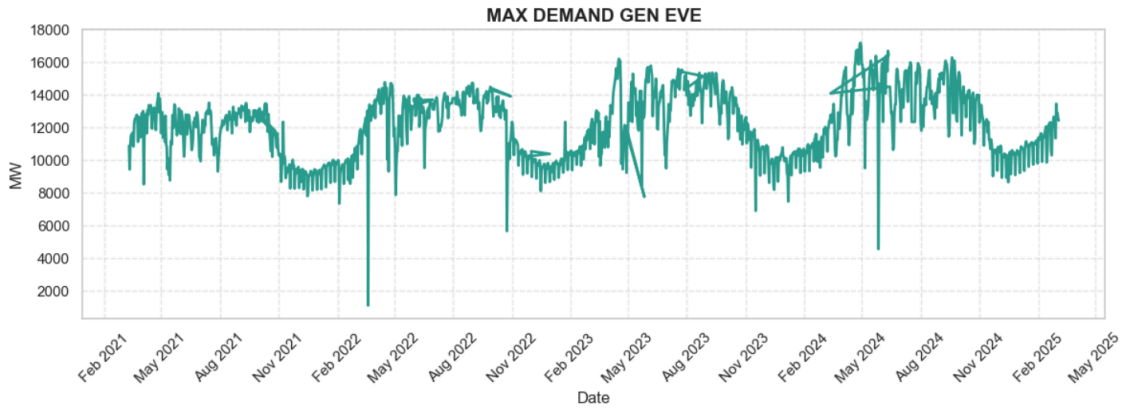


Figure 4.5: Daily Maximum Electricity Demand (March 2021 – March 2025)

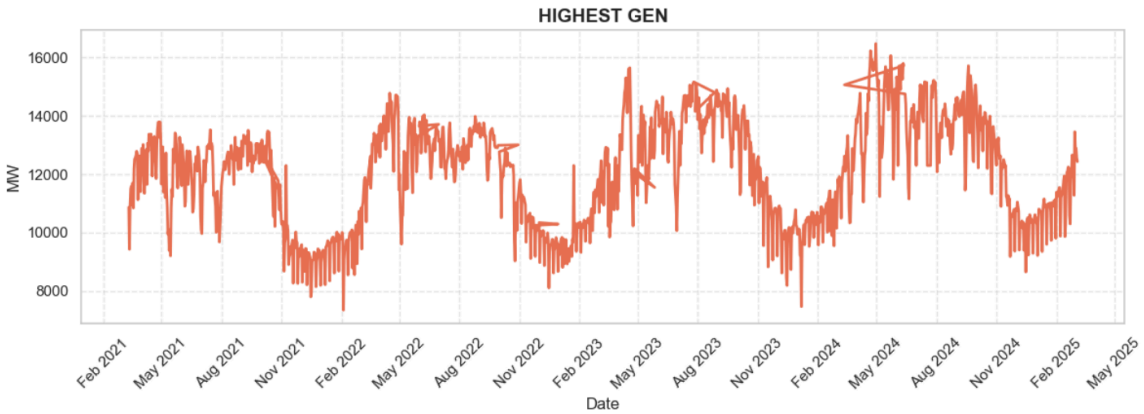


Figure 4.6: Daily Highest Electricity Generation (March 2021 – March 2025)

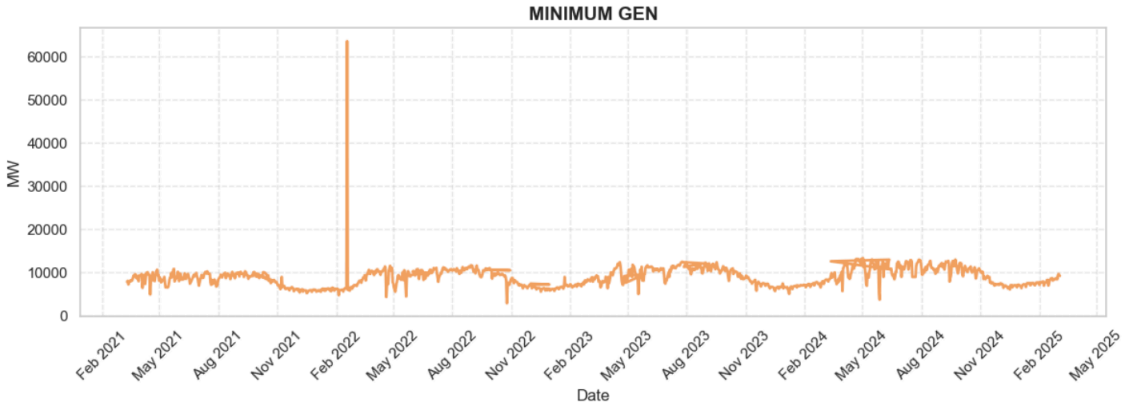
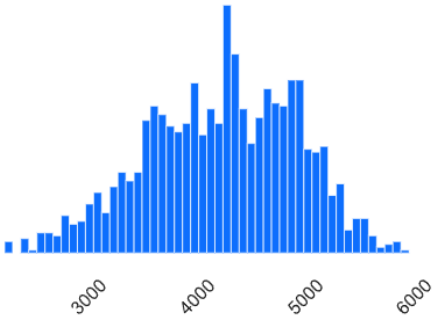


Figure 4.7: Daily Minimum Electricity Generation

- **Dhaka had the Highest Demand:** Among all regions, As shown **Fig. 4.8**, **Dhaka** consistently had the highest electricity demand, with an average value of **4214 MW**. This made Dhaka a critical zone for our forecasting model.



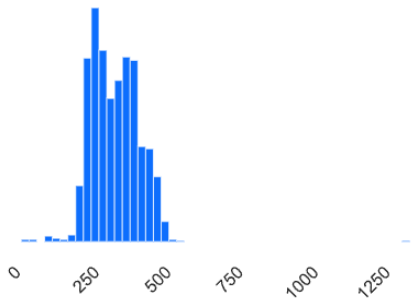


Fig. 4.8: Dhaka Evening Electricity Demand **Fig. 4.9:** Barisal Evening Electricity Demand

- **Barishal Had the Lowest Demand:** In contrast, Barisal recorded the lowest average electricity demand among all zones (see **Fig. 4.9**), with a mean value of approximately **347 MW**. This indicates that Barisal is a relatively low-demand region in the national power distribution network.
- **No Missing Data After Preprocessing:** After filling missing values using column-wise medians, the final dataset used for modeling had **0% missing cells**, ensuring reliability in training and evaluation.
- **Low Number of Outliers:** Most numeric columns had stable distributions with minimal outliers, making the dataset suitable for regression-based models without extensive outlier handling.
- **Interaction Between Maximum Demand and Load Shedding:** The Hexbin (2D histogram plot) plot **Fig. 4.10** shows the relationship between **maximum daily electricity generation** (Max_Demand_Gen_Eve) and **total load shedding** (Total_Load_Shed). Each hexagon represents the density of data points, darker blue areas indicate a higher concentration of records. From the **Fig. 4.10** We can see that

most of the data points are clustered in the **demand range of 10,000 - 14,000 MW**, where **load shedding is relatively low or close to zero**. However, as maximum demand increases beyond **14,000 MW**, there is a visible rise in the spread of load shedding, suggesting that **very high demand can exceed available supply**, leading to more frequent or severe outages. The Fact that **dense clusters remain near zero load shedding** at moderate demand levels implies that **normal operations are generally manageable** within a certain generation capacity.

This interaction highlights how critical it is to balance demand and supply effectively, especially during peak periods.

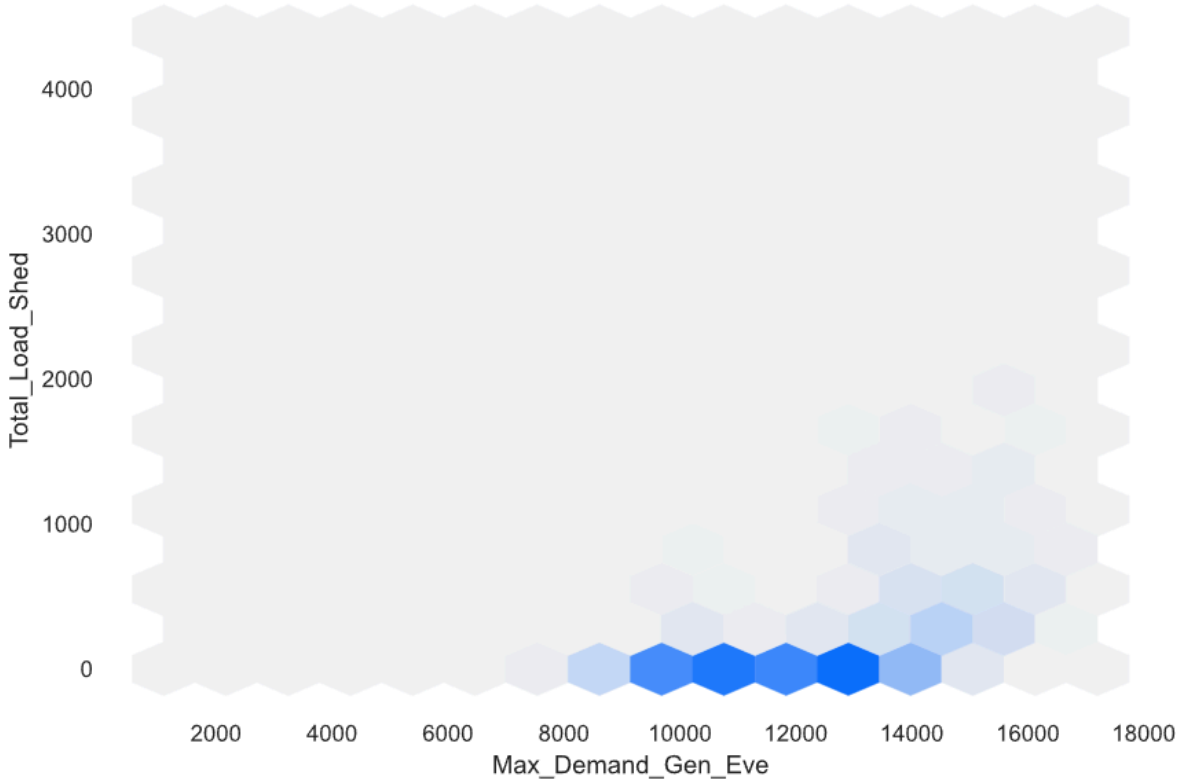


Figure 4.10: Interaction Between Maximum Demand Generation and Total Load Shedding

A correlation matrix of the main features was generated to understand the relationship between input variables (**Figure 4.11**). Strong correlations were observed among time-based features (such as day, month, and weekday) and lagged demand values, especially with rolling averages.

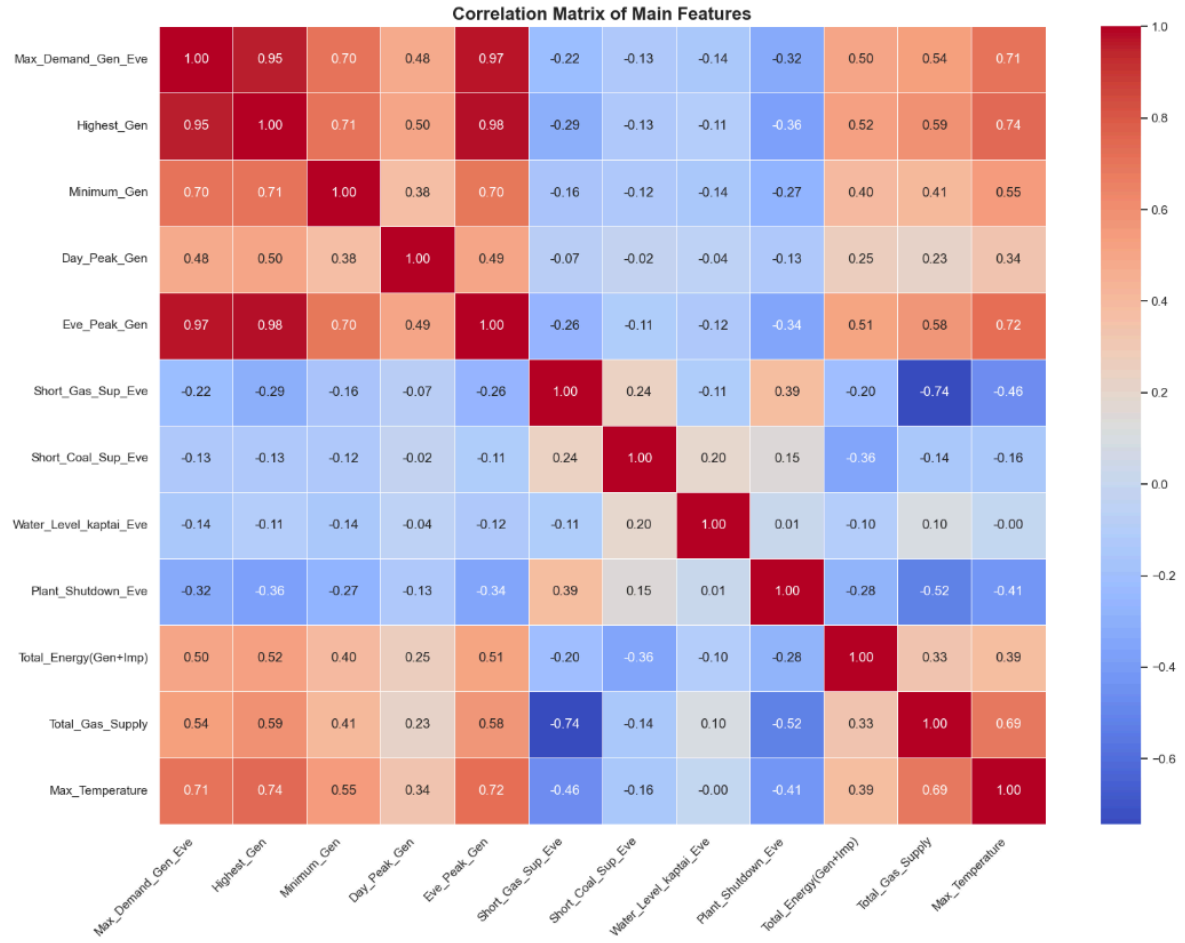


Figure 4.11: Correlation Matrix of Main Features

4.4 Feature Engineering

In this step, we transformed and enriched our dataset to create meaningful inputs (features) that could help machine learning models learn more effectively. Our primary goal was to predict the **evening electricity demand** for each zone in Bangladesh. These demand values were treated as our target variables.

Defining target Variables: We selected the following columns as target variables, one for each zone, representing evening demand in megawatts (MW). Dhaka_Dem_Eve, Chattogram_Dem_Eve, Khulna_Dem_Eve, Rajshahi_Dem_Eve, Cumilla_Dem_Eve, Mymensingh_Dem_Eve, Sylhet_Dem_Eve, Barishal_Dem_Eve, and Rangpur_Dem_Eve.

Dropping Unnecessary Columns: To avoid data leakage (Data leakage in machine learning occurs when a model uses information during training that wouldn't be available at the time of prediction [9]), we dropped columns related to evening supply, load shedding, and total demand/supply values. These columns, if used, could directly reveal the target and reduce model generalization.

Time-Based Features: We added new columns derived from the date index to help the model capture seasonal and temporal patterns, such as day of the month, month, weekday and weekend indicator (Friday and Saturday were marked as weekends based on the Bangladeshi calendar)

Lag and Rolling Features: In time series forecasting, **Lag** means looking at past data points to help predict future ones. It's like using yesterday's temperature to guess today's weather. For example, if we're forecasting monthly sales, a lag of 1 would use last month's sales, and a lag of 2 would look two months back. Lag helps find patterns and trends by showing how past values relate to current ones. It's especially useful in models like ARIMA or machine learning approaches where past data becomes a key input for predictions [10].

Rolling is like sliding a small window over a long sequence of time series data to create multiple smaller pieces (window). Instead of using the entire time series as one chunk, rolling breaks it down into overlapping segments, each ending a bit later than the previous one. This is useful for training forecasting models because it trunks one big sequence into many learning examples. It's especially helpful when using features-based approaches, like with the **tsfresh** library, to make the most out of historical patterns [11].

To capture recent trends in demand, we created **lagged features** (yesterday's demand) and **rolling averages** over the past 3 and 7 days for each zone. These help the model understand short-term consumption behavior.

Examples of generated features:

- **Dhaka_Dem_Eve_lag1** - (yesterday's demand in Dhaka)
- **Chattogram_Dem_Eve_rolling3** - (3-days average demand in Chattogram)
- **Sylhet_Dem_Eve_rolling7** - (7-days average demand in Sylhet)

Final Features: After engineering these features, we dropped rows containing Nan values (introduced due to rolling and lag calculations). This left us with **1444 cleaned and enriched records** and **73 total columns**, out of which **43 were selected as features** for training our model.

How were the 73 columns generated?

The column count resulted from a combination of original variables, time-based features, and newly engineered features:

- **42 numeric columns:** These include the original continuous variables (see **Fig. 4.4**).
- **4 time-based features:** Extracted from the timestamp - **day**, **month**, **weekday**, and **is_weeked**.

  Subtotal: 42 + 4 = 46 columns

- **For each of the 9 target variables, we engineered the following:**
  1. 1 lag feature (*_lag1)
  2. 2 rolling features (*_rolling3, *_rolling7)

  This results in **3 new features** per target x **9 targets** = **27 new features**

Total column count: 46 (original + time-based) + 27 (engineered) = 73 columns

Why did the number of records reduce to 1444?

Originally, the dataset contained **1461 daily records** (see **Fig. 4.4**), representing data from **March 2021 to March 2025** (4 years).

However, the use of rolling windows and lag introduced missing values (**NaNs**) at the beginning of the dataset:

- A 7-day rolling window cause missing values for the first 6 days
- A 1-day lag adds one more missing value
- Due to overlapping effects from multiple rolling and lag operations, the number of rows affected slightly increased.

To ensure data quality, we dropped these rows using **df.dropna(inplace=True)** command. As a result:

- 17 rows with missing values were removed

Final record count: **1461 (original rows)** - **17 (removed)** = **1444 records**

To explore external factors influencing electricity demand, we analyzed the relationship between weather and total energy usage. One notable correlation was observed between **maximum temperature** and **total energy demand**. As shown in **Fig.4.12** , there is a clear upward trend indicating that higher temperatures often correspond with increased energy demand, likely due to air conditioning usage during hotter days. Based on this insight, we include temperature as a relevant feature in our dataset to help the model capture weather-related demand fluctuations.

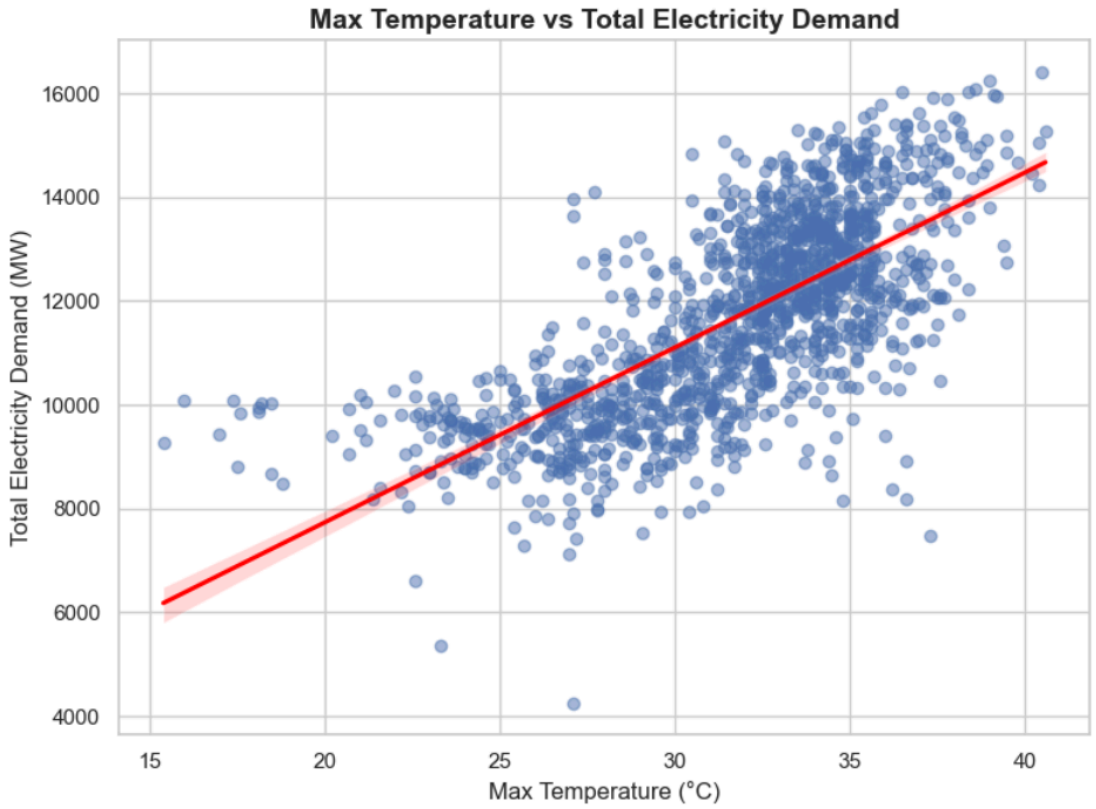


Fig.4.12: Relationship between Maximum Temperature and Total Energy Demand (March 2021 – March 2025)

To better understand which features contribute most to the demand prediction of individual zones, we computed zone-wise feature correlations (**Figure 4.13**). These insights guided the inclusion of specific lag and weather features tailored for each zone.

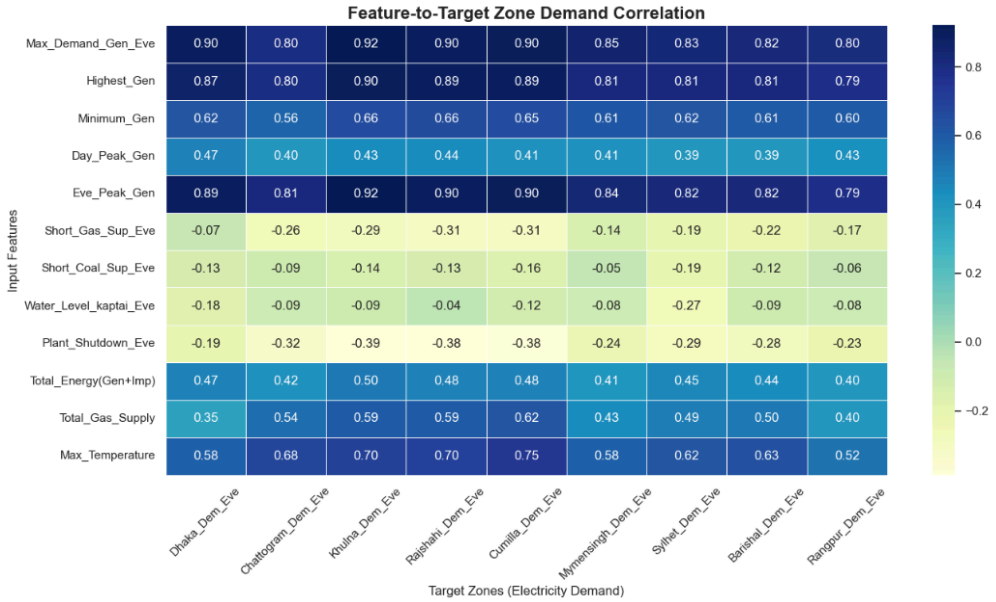


Figure 4.13: Correlation Between Features and Each Target Zone Demand

4.5 Dataset Splitting and Feature Scaling

Before training any machine learning (ML) model, it's essential to evaluate how well it will perform on data it has never seen before. This is where the **Train-Test Split** technique comes in. It allows us to simulate a real-world scenario by dividing the dataset into two parts:

- **Training Set:** used to teach the model patterns and relationships in the data.
- **Testing Set:** Held back and used to evaluate how well the model performs on unseen data.[12]

In our project, we used a time-based split strategy instead of random splitting. Since electricity demand data is time-series in nature (ordered by data), random splitting could lead to data leakage [9] from future to past. To prevent this, we split the dataset chronologically. The first 80% of the records were used for training, and the remaining 20% for testing.

After the split, we had:

X_train: 1155 records, 43 features, **X_test:** 289 records, 43 features, **y_train:** 1155 records, 9 target columns (zone-wise electricity demand), **y_test:** 289 records, 9 target columns.

This method ensures our model learns from past data and is tested on future-like data, making the evaluation more realistic and robust.

Before training any machine learning model, it's important to ensure that all input features are on a similar scale. Since many algorithms (like K-Nearest Neighbors, SVM, and even gradient-based methods) are sensitive to the range of input data, proper scaling helps models converge faster and perform better[13].

In our project, we used **MinMaxScaler** from the **sklearn.preprocessing** module. This technique scales each feature individually so that it falls within a specified range between **0 and 1(see Figure 4.14)**. The formula used by **MinMaxScaler** is:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

This transformation preserves the shape of the original distribution while bounding all features within the target range. It's especially useful when we want to retain the sparsity of the dataset or when dealing with features with known minimum and maximum boundaries.

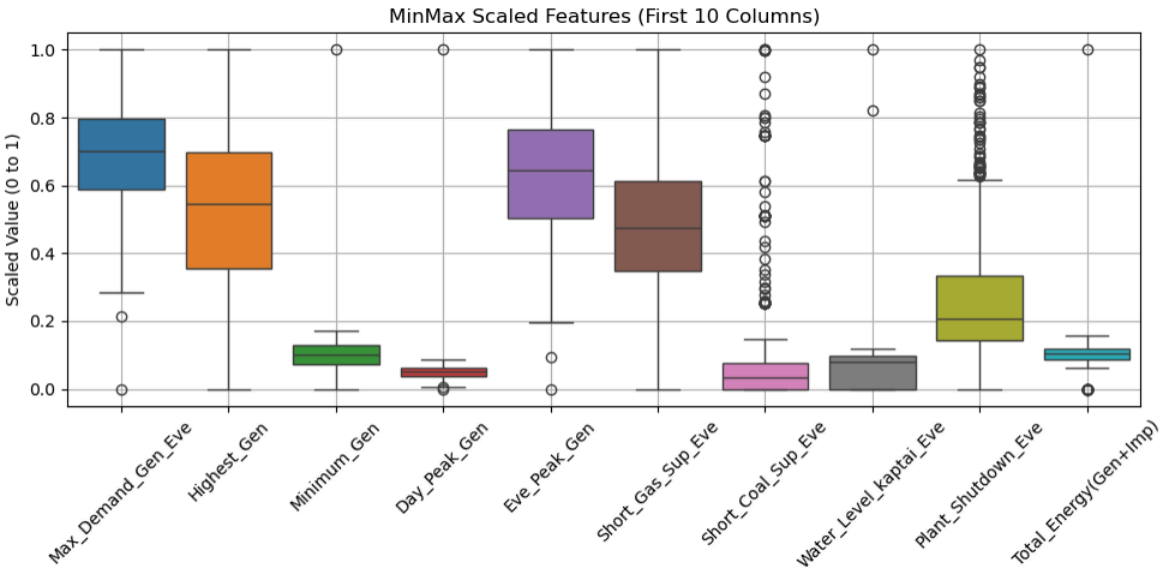


Figure 4.14: Visualization of Feature Values After MinMax Scaling (First 10 Features)

4.6 Model Training and Evaluation

1. Linear Regression

Linear Regression is one of the simplest and most interpretable machine learning algorithms. It models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. The goal is to find the best-fit line that minimizes the residual sum of squares between the observed and predicted values [14].

Linear regression is a statistical method used to model the relationship between a dependent variable and one more independent variable. It provides valuable insights for prediction and data analysis [15].

Equation:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Where, θ0 is the intercept and θ1 is the slope.

To find the best-fitting line that minimizes the prediction error, measured by the **Mean Squared Error (MSE)**.

Assumptions: Linearity between variables, No multicollinearity, Homoscedasticity, Normal distribution of residuals, Independence of observations.

Evaluation Metrics: R^2 Score (How well the model explains the variance in the data), MAE/RMSE (Measure of prediction error).

Limitation: Performs poorly when relationships are non-linear or when features are highly correlated.

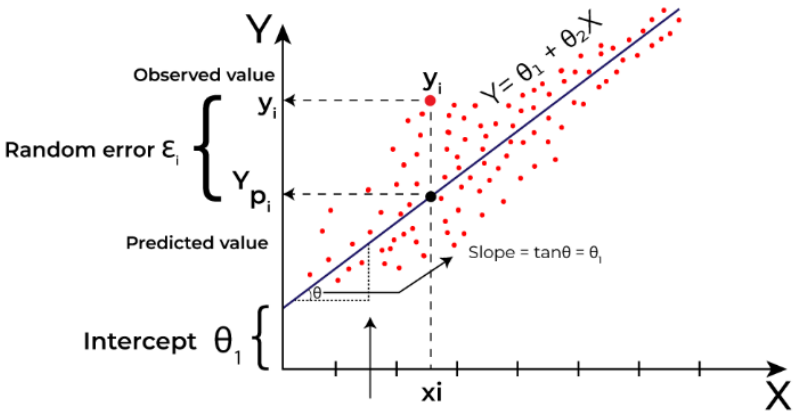


Figure 4.15: Linear Regression

We applied Linear Regression using **MultiOutputRegressor** to predict the evening electricity demand across all 9 zones simultaneously. This approach works well when the target variables (zone-wise demands) have some correlation and the relationship between features and target is approximately linear (dataset has continuous values). While it served as a strong baseline model, performance was limited by the algorithm's inability to capture complex, non-linear patterns in the data.

2. Decision Tree Regressor

Decision Tree Regression is a powerful and intuitive machine learning technique.Unlike traditional linear regression, which assumes a straight-line relationship between input features and the target variable, Decision Tree Regression is a non-linear regression method that can handle complex datasets with intricate patterns. It uses a tree-like model to make predictions, making it both flexible and easy to interpret [16].

How it Works: It builds a **tree-like structure**, where

- **Decision nodes** ask questions (e.g., “Is temperature > 25?”)
- **Leaf nodes** hold prediction values (e.g., average demand of that node’s data)

The algorithm split the dataset at each node to **minimize prediction error,** usually by reducing the **Mean Squared Error (MSE)**.

Advantages: Can handle non-linear relationships, Easy to visualize and interpret, requires little data preparation (no need for features scaling or normalization).

Limitations: Overfitting is common if the tree is not pruned and can be unstable to small in data.

Real-World Use Case (example from article): Predict house prices by splitting based on **location** → **size** → **age** accurately estimate the price.

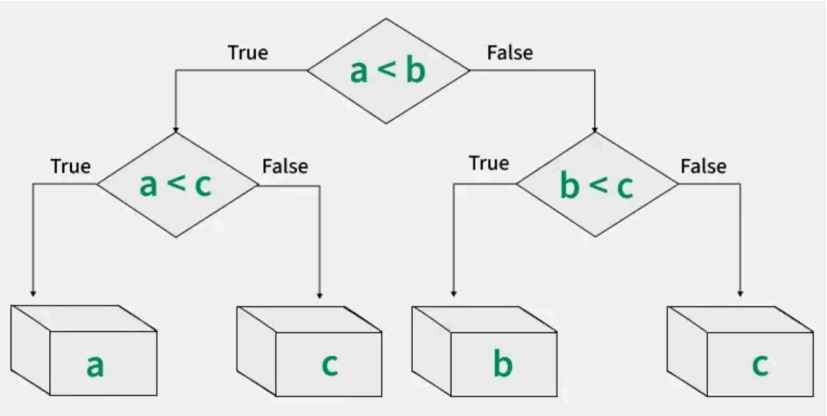


Figure 4.16: Workflow of Decision Tree Regressor

We implemented a **DecisionTreeRegressor** wrapped in **MultiOutputRegressor** to model zone-wise electricity demand. The decision tree model captured nonlinear relationships and interactions between features, such as weather trends and historical electricity usage. However, it tended to overfit the training data due to its hierarchical nature, especially when not pruned or regularized.

3. Random Forest Regressor

Random Forest Regression is an ensemble machine learning algorithm used for predicting continuous numeric values. It works by combining predictions from multiple Decision Trees to improve accuracy and reduce overfitting [17].

How it Works:

- **Bootstrap Sampling:** Random samples (with replacement) are drawn to train each tree.
- **Feature Sampling:** Each tree uses a random subset of features to increase model diversity.
- Aggregation: For regression tasks, the final output is the average of all individual tree predictions.

Why It's Powerful

- Reduces overfitting compared to a single Decision Tree.
- Handles non-linear relationships and interactions between features.
- Offers better generalization on unseen data.
- Provides features and important metrics for interpretation.

Scikit-learn Implementation

- **RandomForestRegressor()** from **sklearn.ensemble** is used to implement this model.
- Other useful tools in the workflow:
    1. **train_test_split()** - splitting data
    2. **StandardScaler()** - feature scaling
    3. **cross_val_score()** - performance evaluation via k-fold cross-validation

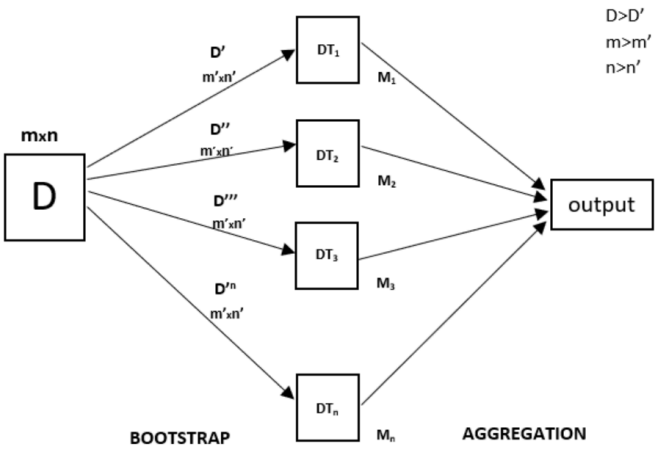


Figure 4.17: Random Forest Regression Model Working

Using **RandomForestRegressor** in a multi-output setting helped us achieve more stable and accurate predictions compared to a single decision tree. It handled feature interactions and reduced model variance by averaging results over numerous trees. This made it a strong candidate for modeling the temporal and spatial complexity of our dataset.

4. AdaBoostRegressor

AdaBoost (short for Adaptive Boosting) is a powerful **ensemble learning algorithm** that builds a **strong model** by combining multiple **weak learners**, typically simple models like decision stumps (shallow decision trees). It focuses on **correcting the errors** made by earlier models [18].

How It Works

- **Sequential Learning:** Each new model is trained on data that was misclassified by the previous one.
- **Weighting:** Misclassified points are given **higher weights**, making them more important for the nest model.

- **Final Prediction:** Models are combined using a **weighted majority vote** (classification) or **weighted average** (regression), where better models have more influence.

Key Characteristics: Learn from mistakes iteratively, Works well even with **simple base learner**, **Reduces bias and variance**, improves model performance, Sensitive to **noisy data and outliers**, as it gives more weight to hard-to-predict samples.

Python Implementation: The Adaboost algorithm can be implemented manually for better conceptual understanding. For practical applications, **AdaBoostRegressor** from **sklearn.ensemble** is widely used. Common base estimator (**DecisionTreeRegressor**, max_depth=1).

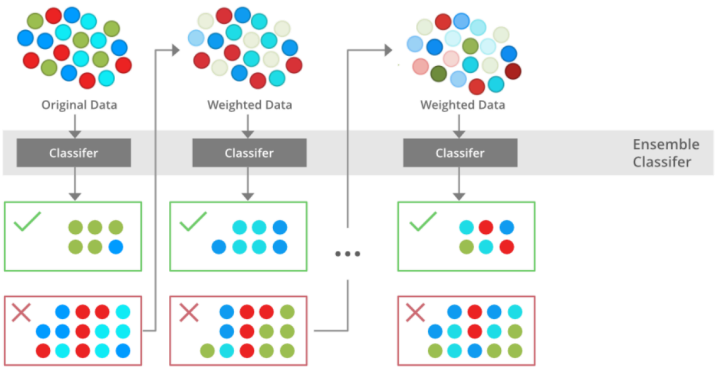


Figure 4.18: Boosting Algorithm

The AdaBoostRegressor was applied using shallow decision trees for each zone's prediction. It improved performance on harder-to-predict zones, especially those with more fluctuation like Chattogram and Sylhet. However, due to the sequential training of weak learners, the model was sensitive to noise and outliers in some zones.

5. XGBoost Regressor

XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of gradient boosting algorithms. It is an ensemble learning method that builds **multiple models** (weak learners) and combines them to produce a more accurate robust prediction.

In regression tasks, XGBoost predicts continuous outcomes, making it ideal for problems like choice pricing, sales forecasting, or electricity demand prediction [19].

Core Principles

- **Objective Function:** Consists of two parts:
    1. **Loss Function (e.g., RMSE, MSE):** Measures how far predicted values are from actual values.
    2. **Regularization Term:** Controls model complexity to reduce overfitting.
- **Base Learners:** Uses decision trees or linear models as weak learners. These learners are combined in a stage-wise fashion using gradient boosting.

- **DMatrix:** A specialized data structure that optimises memory and computation performance. Required by XGBoost for training and prediction.

Evaluation Metrics:

- **RMSE (Root Mean Squared Error):** Measure standard deviation of prediction error.
- **MSE (Mean Squared Error):** Average of squared differences between actual and predicted values.
- **MAE (Mean Absolute Error):** Used less frequently due to lack of sensitivity to larger errors.

Why Use XGBoost: It improves accuracy by focusing on errors made by previous models, Incorporates regularization to enhance generalization, Offers parallel computation, tree printing, and missing value handling for better efficiency.

We used XGBRegressor with MultiOutputRegressor to train one model per zone. XGBoost outperformed most other individual models due to its ability to handle missing values, features interactions, and non-linear trends. It consistently produced high R^2 scores and low RMSE values across almost all zones, making it one of our top-performing algorithms.

6. Support Vector Regressor (SVR)

Support Vector Regression (SVR) is a regression algorithm based on **Support Vector Machines (SVMs)**. Unlike standard regression, SVR attempts to fit a function within a margin of tolerance $\pm\varepsilon$ (epsilon) from the actual data points, balancing prediction accuracy with model simplicity. SVR is built upon the same principles as Support Vector Machines (SVM), focusing on margin maximisation and support vectors. SVR tries to keep predicting within a tube of width $\pm\varepsilon$ from the actual values, ignoring errors within this range. For Loss Function, errors outside this margin are penalized via a hinge-like loss function. Only data points outside the epsilon-tube contribute to the model (become support vectors) [20].

Kernels: Linear Kernel (fits straight-line relationships), **RBF (Non-Linear) Kernel** (Captures complex patterns).

Model Evaluation: Split data → train and test sets, Use **MSE**, **RMSE**, **MAE**, and **R^2** for evaluation.

SVR was used to model each zone's demand by wrapping it inside a MultiOutputRegressor. It showed moderate performance, performing better in zones with smoother demand curves but struggling with highly fluctuating ones. SVR also required careful feature scaling (MinMax) due to its sensitivity to input magnitudes.

7. K-Nearest Neighbors Regressor (KNN)

KNN regression is a non-parametric method used for predicting continuous values. The core idea is to predict the target value for a new data point by averaging the target value of the K-Nearest Neighbors in the feature space. The distance between data points is typically measured using Euclidean distance, although other distance metrics can be used [21].

How It Works: Choose K (A small K is sensitive to noise; a larger K gives smoother predictions), **Measure Distance** (Usually Euclidean distance), **Find Neighbors** (Identify K closest data points), **Predict Output** (Use the mean of the neighbors target values).

Model Evaluation: Mean Squared Error (Measures average squared difference), **R^2 Score** (Indicates the proportion of variance explained by the model).

We applied **KNN** for each zone using **MultiOutputRegressor**. While it worked well for stable zones with consistent patterns (e.g., Barisal), it was computationally expensive and performed poorly when there was high variability or when similar historical cases were limited.

8. Ensemble Models

To further improve the accuracy and robustness of zone-wise electricity demand predictions, we implemented three ensemble techniques: **Voting Regressor**, **Stacking Regressor**, and **Custom Averaging**. These ensemble methods leverage the strengths of multiple base models to generate more stable and generalized predictions [22].

8.1 Voting Regressor

Voting Regressor is an ensemble learning method used for regression tasks that combines the predictions from multiple individual models to improve overall performance. It works either by simple averaging or weighted averaging of the output values from its base estimators, thereby reducing prediction variance and increasing robustness [23].

We used a combination of high-performing models, such as **Random Forest**, **XGBoost**, and **Linear Regression** in a Voting Regressor setup. The average prediction helped reduce **overfitting** and produced consistently better **R^2** and **RMSE** azores across most zones. This method worked particularly well when base models had uncorrelated errors.

8.2 Stacking Regressor

Stacking regressor (or Stacked Generalization) is another powerful ensemble technique where multiple base models are trained to solve the same problem, and their predictions are then used as input features for a second-level model (meta-learner). This meta-learner learns how to best combine the base models outputs, often resulting in better performance than any single model [24].

We implemented stacking using **Random Forest**, **XGBoost**, and **Linear Regression** as the meta-learner. This approach captured complex relationships between model predictions and delivered superior performance, especially in zones with non-linear demand patterns. It was one of the most accurate models in terms of both **RMSE** and **R²** scores.

8.3 Custom Averaging

Custom Averaging is a simplified ensemble approach where predictions from selected base models are averaged manually, usually based on their past performance or reliability.

We selected the top 3 models (XGBoost, Random Forest and Linear Regression) based on validation results and averaged their predictions to form a custom ensemble. This method, although heuristic, helped improve stability and was computationally lighter than stacking. It showed good performance consistency across both high-demand zones like Dhaka and lower-demand zones like Barisal.

4.7 Result Interpretation and Model Comparison

This section presents a comprehensive analysis of the performance of all regression models implemented in this study. The evaluation is based on three standard regression metrics: **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R²)**. Additionally, a zone-wise comparison is provided to assess the prediction consistency across different geographic regions.

Evaluation Metrics Overview

To assess model effectiveness, the following metrics were used:

- **Mean Absolute Error (MAE):** Measure the average magnitude of prediction errors, without considering their direction.
- **Root Mean Squared Error (RMSE):** Similar to MAE but penalizes larger errors more heavily, making it sensitive to outliers.
- **R-Squared (R²):** Represents the proportion of variance in the target variable that is explained by the model. An R² closer to 1 indicates better performance.

These metrics help provide both an intuitive and statistical measure of the prediction accuracy.

Overall Model Performance

The table below summarizes the overall evaluation metrics for each regression model.

Model	MAE	RMSE	R^2 Score
Linear Regression	37.41	59.63	0.89
Random Forest	48.41	81.69	0.83
XGBoost	50.14	83.14	0.81
AdaBoost	57.38	90.53	0.78
KNN	82.30	136.51	0.59
Decision Tree	85.72	148.24	0.50
SVR	121.52	238.30	0.34

Table 4.1: Overall Model Performance

Interpretation:

- **Linear Regression** outperformed all other models across all metrics. Despite its simplicity, it successfully captured the underlying trends in the data.
- **Random Forest** and **XGBoost** delivered robust performances, benefiting from their ensemble structure and ability to model non-linear relationships.
- **AdaBoost** showed moderate results, better than the base learners like Decision Tree and KNN, but behind Random Forest and XGBoost.
- **KNN**, **Decision Tree**, and especially **SVR** struggled with accuracy, likely due to overfitting or poor handling of high-dimensional multi-output data.

Zone-wise Performance Comparison

To further evaluate the model's robustness, a zone-wise performance analysis was conducted based on the R^2 score for each geographic zone. The **Zone-wise Model Performance Heatmap** (**Figure 6.1**) clearly visualizes this comparison.

Key Observations:

- **Linear Regression** demonstrated consistent accuracy across all zones, with R^2 scores exceeding 0.82 in every region. It performed exceptionally well in **Dhaka(R^2=0.95)**, **Comilla (0.94)**, and **Khulna (0.93)**.
- **Random Forest** and **XGBoost** showed strong performance in **Khulna**, **Comilla**, and **Rajshahi**, proving their effectiveness for complex regional patterns.
- **AdaBoost** achieved moderate zone-wise accuracy but did not outperform ensemble counterparts in any specific zone.
- **SVR** performed poorly across all zones, with **Dhaka** scoring as low as **R^2 = -0.45**, highlighting its limitations in this context.
- **KNN** delivered slightly better results than SVR, but still lacked the accuracy and consistency required for reliable predictions.

Figure 4.19: Zone-wise Heatmap of R^2 Score Across All Models *(A visual heatmap showing model performance by zone, with darker shades indicating higher R^2)*

This visualization reinforces the superiority of simpler and ensemble-based models over more complex or instance-based regressors on this specific dataset.

Ensemble Model Comparison

To enhance predictive performance further, ensemble strategies such as **Voting Regressor**, **Stacking Regressor**, and a **Custom Averaging Ensemble** were implemented.

Zone	Stacking	Voting	Custom Avg
Dhaka	0.9361	0.9399	0.9399
Chattogram	0.8203	0.8194	0.8194
Khulna	0.9167	0.9262	0.9262
Rajshahi	0.8867	0.8786	0.8786
Comilla	0.9312	0.9364	0.9364
Mymensingh	0.8641	0.8259	0.8259
Sylhet	0.8625	0.8591	0.8591
Barisal	0.8741	0.8689	0.8689
Rangpur	0.8150	0.8094	0.8094

Table 4.2: Ensemble Model Performance by Zones

Interpretation:

- **Voting Regressor** and **Custom Averaging** achieved the highest R^2 scores in most zones, confirming their strong generalization capability.
- **Stacking Regressor** also performed well but was marginally outperformed in several zones.
- All ensemble methods consistently outperformed base models, confirming the effectiveness of model combination in improving prediction accuracy.

Summary of Findings

To conclude the model evaluation, this section highlights the most significant insights delivered from both numerical results and visual analysis.

- **Best Overall Model:** Among all individual models, **Linear Regression** delivered the highest performance across all evaluation metrics (MAE, RMSE, and R^2) (**See Figure 4.20**). Its simplicity and ability to generalize made it the most reliable choice for this dataset. The model consistently achieved R^2 azores above 0.89 in all zones, with top performance in **Dhaka (R^2 = 0.95)**, **Comilla (0.94)**, and **Khulna (0.93)**.

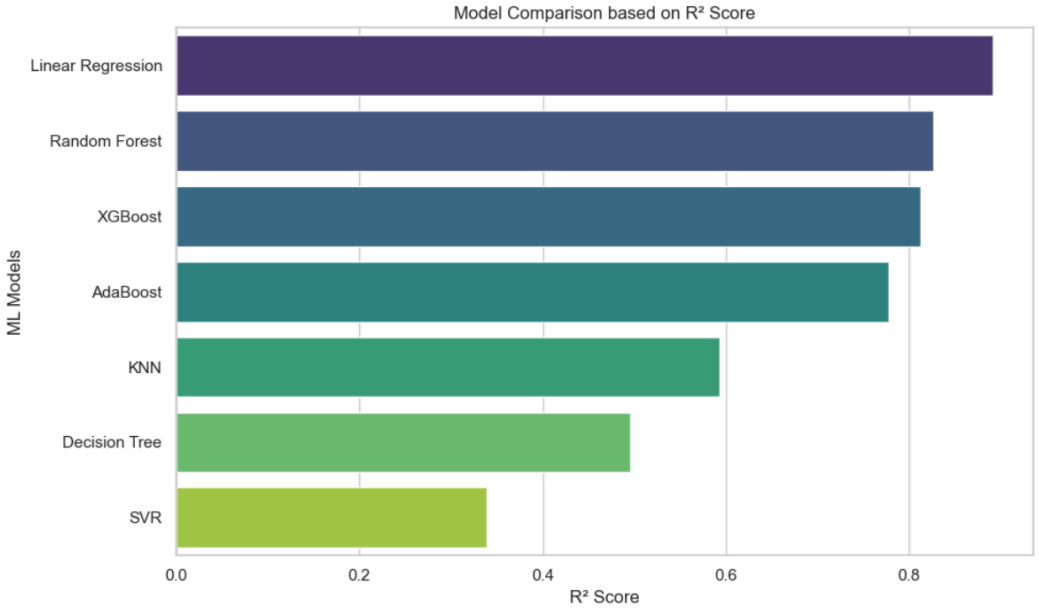


Figure 4.20: Model Comparison based on R^2 Score

- **Most Consistent Models Across Zones:** Random Forest and XGBoost maintain solid R^2 scores (Typically above 0.80) across all zones, providing effectiveness in handling regional variability and nonlinear dependencies in electricity demand. (**See Figure 6.2**)
- **Weakest Performing Models: Support Vector Regression (SVR)** and **Decision Tree Regressor** produced the lowest performance metrics, including negative R^2 scores in some zones (**SVR in Dhaka: R^2 = -0.45**). These models struggled with multi-output, high-dimensional time-series data, and were highly sensitive to outliers.

- **High Accuracy Zones:** The **Dhaka zone** consistently achieved the highest prediction accuracy across nearly all models, likely due to its more regular demand patterns (**see Figure 4.21)**and higher-quality data. **Comilla** and **Khulna** also showed high predictability.

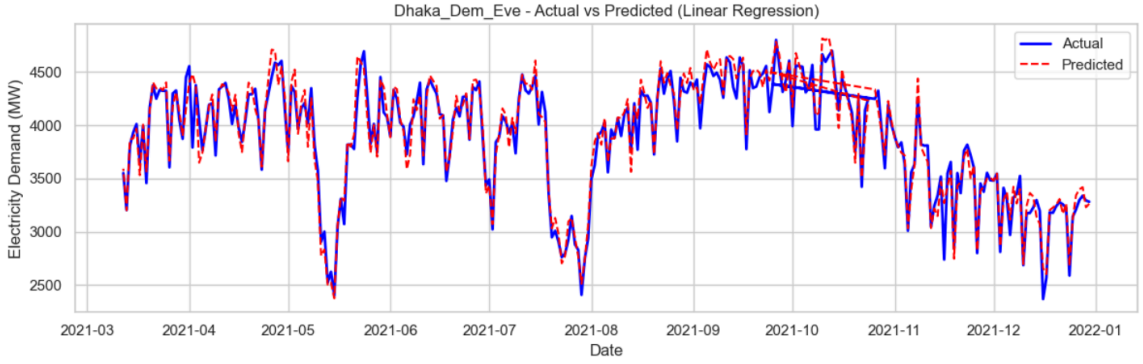


Figure 4.21: Dhaka demand at Evening (Actual vs Predicted)

4.8 Model Interpretation

As machine learning is increasingly used in critical sectors like healthcare, finance, and energy, the need for models to be not just accurate but also interpretable has become essential. This has led to the rise of **Explainable Artificial Intelligence (XAI)**, a field focused on making complex models understandable to humans. While powerful models like ensembles and deep learning offer high accuracy, they often act as "**black boxes**," making it difficult to trust or validate their predictions [GeeksforGeeks, 2025].

One widely used XAI method is **LIME (Local Interpretable Model_agnostic Explanations)**. LIME explains individual predictions by approximating the complex model locally with a simpler, interpretable model such as linear regression. It highlights which features influenced the prediction and whether their effects were positive or negative [25]. **LIME** is especially useful in applications like electricity demand forecasting, where understanding why a model made a certain prediction for a specific zone helps build trust and inform better decisions.

LIME in Our Project: Zone-Wise Electricity Demand Prediction

In our project, we used **LIME** to interpret the predictions made by our best-performing ML model for zone-wise evening electricity demand prediction. The LIME package in Python helped us visualize which features influenced the predictions the most for individual zones. Below, we explain key insights for selected zones:

Dhaka's Evening Demand (Dhaka_Dem_Eve)

- **Predicted Demand:** 3196.37 (within a possible range of 2644.41 to 5621.99)
- **Top Positive Influencers:** Dhaka_Dem_Eve_rolling3(+831.31), Max_Demand_Gen_Eve (+414.13), Eve_Peak_Gen (+285.14).

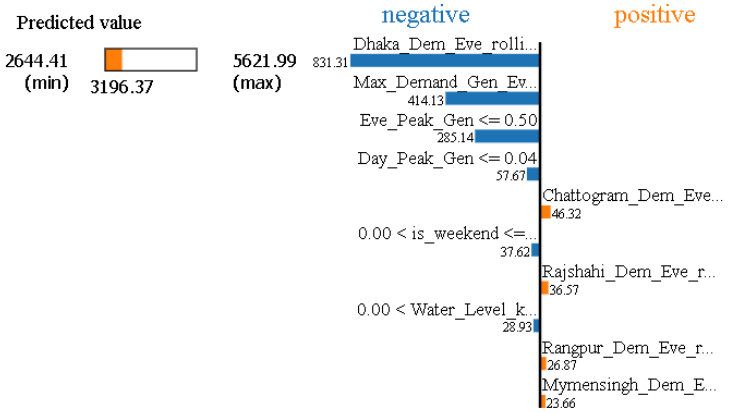


Figure 4.22: Explaining Prediction for Dhaka

Insights: The rolling average of Dhaka's evening demand, coupled with peak and maximum generation values, had a strong positive influence, showing how recent trend and supply availability significantly impacted the prediction.

Chattogram Evening Demand (Chattogram_Dem_Eve)

- **Predicted Demand:** 968.49 (within a range of 690.83 to 1470.67)
- **Top Positive Influencers:** Chattogram_Dem_Eve_rolling3 (+309.23), Eve_Peak_Gen (+46.35), Max_Demand_Gen_Eve (+29.17)

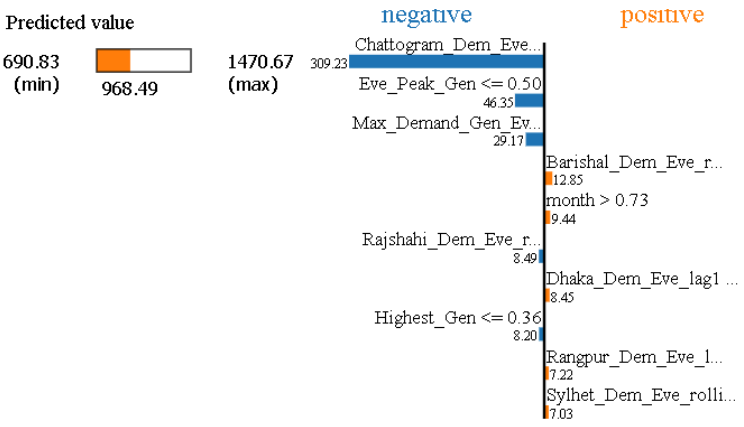


Figure 4.23: Explaining Prediction for Chattogram

Insights: Chattogram's short-term demand history is the strongest signal, while peak demand indicators and past month also have influence.

Khulna Evening Demand (Khulna_Dem_Eve)

- **Predicted Demand:** 979.36 (within 790.32 to 1917.92)
- **Top Influencers:** Khula_Dem_Eve_rolling3 (+326.16), Max_Demand_Gen_Eve (+122.23), Eve_Peak_Gen (+85.82)

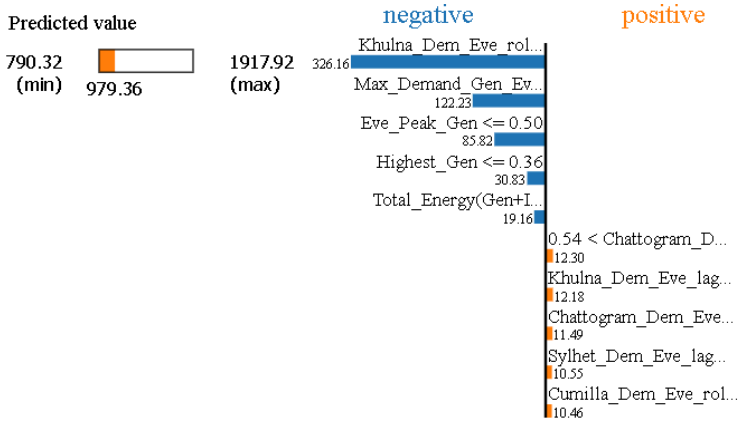


Figure 4.24: Explaining Prediction for Khulna

Insights: Demand history and generation potential strongly drive Khulna's forecast. Lag and generation-related features also support the model's confidence.

Rajshahi Evening Demand (Rajshahi_Dem_Eve)

- **Predicted Demand:** 858.08 (within 773.35 to 1735.98)
- **Top Influencers:** Rajshahi_Dem_Eve_rolling3 (+378.95), Max_Demand_Gen_Eve (+69.40)

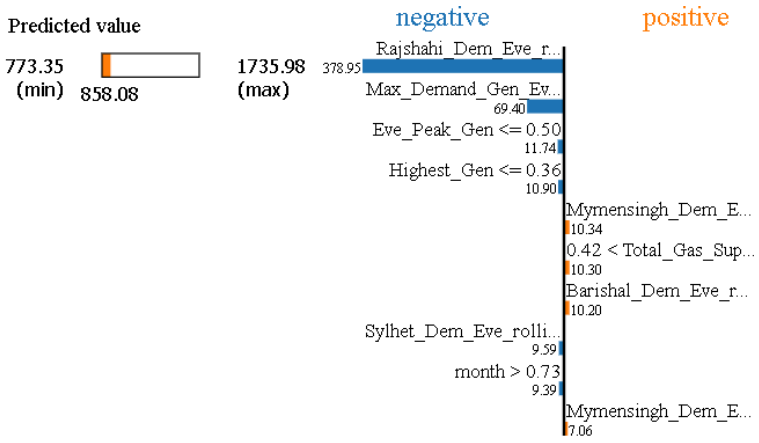


Figure 4.25: Explaining Prediction for Rajshahi

Insights: The model relies heavily on recent local trends and maximum generation figures to make its predictions.

Comilla Evening Demand (Cumilla_Dem_Eve)

- **Predicted Demand:** 712.09 (within 539.32 to 1451.81)
- **Top Influencers:** Cumilla_Dem_Eve_rolling3 (+310.88), Eve_Peak_Gen (+58.63)

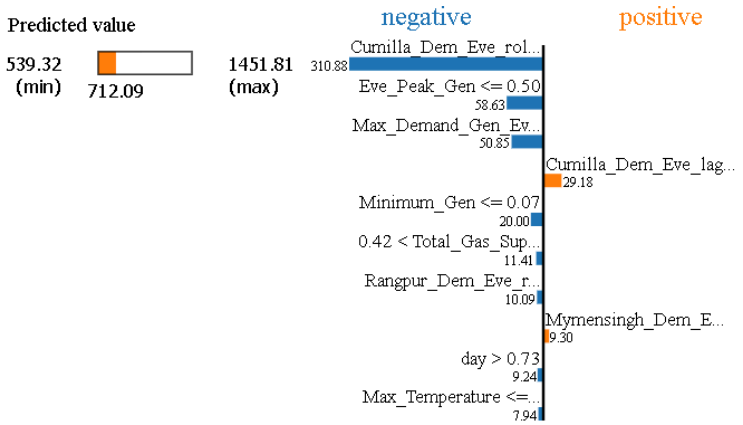


Figure 4.26: Explaining Prediction for Cumilla

Insights: Cumilla's demand forecast was shaped primarily by its rolling average and daily peak generation indicators.

Mymensingh Evening Demand (Mymensingh_Dem_Eve)

- **Predicted Demand:** 702.88 (within 433.36 to 1353.81)
- **Top Influencers:** Mymensingh_Dem_Eve_rolling3 (+329.51), Max_Demand_Gen_Eve (+48.62)

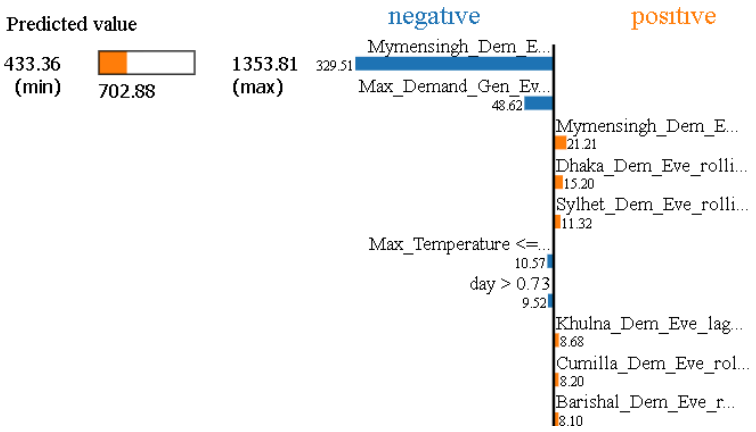


Figure 4.27: Explaining Prediction for Mymensingh

Insights: Mymensingh follows a pattern similar to other zones, with short-term demand history being the dominant feature.

Sylhet Evening Demand (Sylhet_Dem_Eve)

- **Predicted Demand:** 334.19 (within 241.15 to 677.37)
- **Top Influencers:** Sylhet_Dem_Eve_rolling3 (+180.58), Sylhet_Dem_Eve_lag1(+19.80)

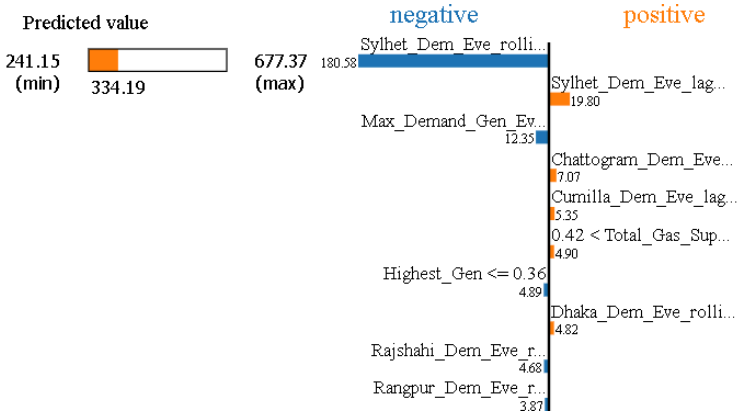


Figure 4.28: Explaining Prediction for Sylhet

Insights: Sylhet's forecast is influenced mainly by immediate past trends and lags, reflecting consistency and recent consumption patterns.

Barisal Evening Demand (Barishal_Dem_Eve)

- **Predicted Demand:** 239.17 (within 164.51 to 656.36)
- **Top Influencers:** Barishal_Dem_Eve_rolling (+145.74), Cumilla_Dem_Eve_rolling (+31.53)

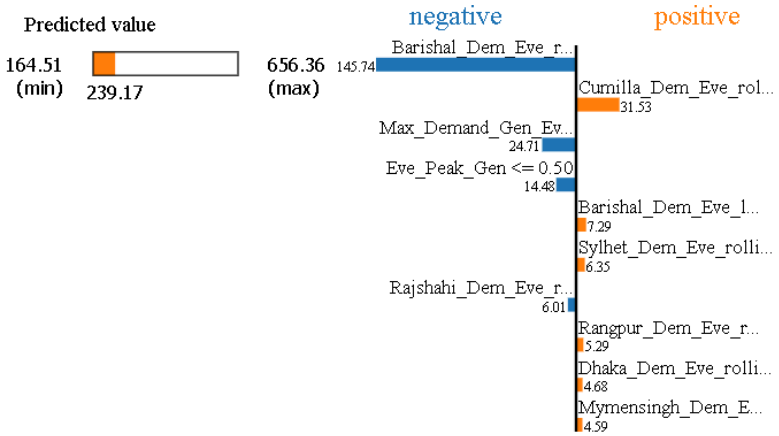


Figure 4.29: Explaining Prediction for Barisal

Insights: Regional interdependencies and local demand history shaped Barishal's forecast, highlighting shared grid behavior.

Rangpur Evening Demand (Rangpur_Dem_Eve)

- **Predicted Demand:** 522.77 (within 599.89 to 1075.71)
- **Top Influencers:** Rangpur_Dem_Eve_rolling (+193.87), Max_Dem_Den_Eve(+19.04)

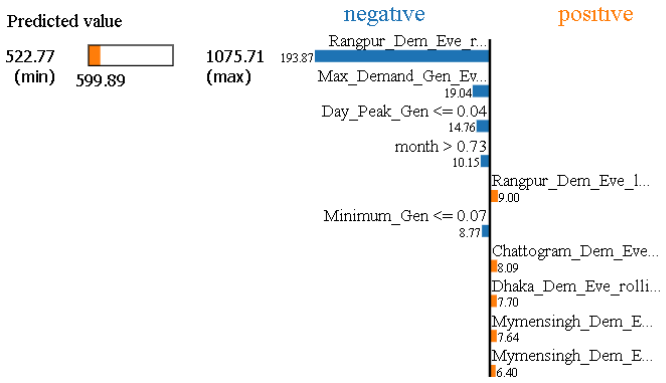


Figure 4.30: Explaining Prediction for Rangpur

Insights: Rangpur's evening demand is primarily driven by short-term trends and previous day's usage, along with seasonal and generation-related factors, reflecting both local consistency and broader grid dynamics.

Using LIME gave us the ability to build trust in our ML model by showing exactly which features affected predictions and how. This is particularly useful for policymakers and engineers in the power sector who are concerned not just with what the predictions are, but why they are so. In a critical domain like electricity forecasting, explainability isn't just a luxury, it is a necessity for operational reliability and public accountability.

5. Conclusion

This project aimed to develop an effective and interpretable machine learning framework for predicting zone-wise electricity demand across Bangladesh, using multi-output regression techniques. Through rigorous experimentation, evaluation, and visualization, we were able to compare and contrast the performance of various regression models on both aggregate and regional levels.

Among the individual models, **Linear Regression emerged as the most accurate and consistent performer**, outperforming more complex alternatives like SVR, Decision Tree, and KNN. Surprisingly, its simplicity allowed it to capture demand trends effectively across all zones, particularly in Dhaka, Comilla, and Khulna. Ensemble techniques such as **Voting Regressor** and **Custom Averaging** further improved the performance, confirming that combining multiple models can enhance generalization and predictive reliability.

Zone-wise evaluations revealed insightful regional patterns. Areas with more stable and higher-quality data, like Dhaka and Comilla, showed higher prediction accuracy, while zones with more volatile or inconsistent demand, such as Barisal and Rangpur, posed greater challenges.

To enhance model transparency and trust, we incorporated **LIME** for local interpretability. This helped us unpack the “black box” nature of our predictions and understanding which features most strongly influenced the demand in each zone. Such explainability is not only useful for validating model outputs but also critical for real-world deployment in energy planning and decision-making.

In summary, this project successfully demonstrated that machine learning models, when carefully selected, tuned, and interpreted, can play a significant role in forecasting electricity demand with high accuracy. The insights generated through both performance metrics and interpretability tools provides a strong foundation for future applications in **smart grid management**, **load balancing**, and **energy distribution optimization** in Bangladesh and beyond.

6. Future Work

In the next phase of this project, more advanced time-series forecasting models such as LSTM (Long Short-Term Memory) or Temporal Convolutional Networks (TCN) can be explored to capture complex temporal patterns in electricity demand. These deep learning models are capable of learning long-term dependencies and may offer better performance in zones with irregular or non-linear demand fluctuations. Additionally, experimenting with hybrid models that combine classical and neural approaches could further enhance accuracy and adaptability across different regions.

Another important direction is to extend this work towards l**oad shedding prediction and optimization**. By integrating supply-side data such as fuel shortages, generation capacity, and weather impacts, it is possible to build models that can anticipate load shedding events before they occur. Once these predictions are reliable, optimization techniques like linear programming or reinforcement learning could be applied to generate efficient and fair load shedding schedules. This would not only help in reducing the negative impact on end users but also support smarter and more balanced electricity distribution across the country.

References

[1] Bangladesh Power Development Board (BPDB), *Annual Report 2022-2023*, Available at: http://www.bpdb.gov.bd/

[2] Bangladesh Power Development Board, *Daily Generation Archive*, Available: https://misc.bpdb.gov.bd/daily-generation-archive

[3] Bangladesh Power Development Board (BPDB), *Daily Forecast Reports*, Available: http://www.bpdb.gov.bd/

[4] G. Vijendar Reddy, L. J. Aitha, C. Poojitha, A. N. Shreya, D. K. Reddy, G. S. Meghana, "Electricity Consumption Prediction Using Machine Learning," *E3S Web of Conferences*, vol. 391, 01048, 2023. https://www.e3s-conferences.org/articles/e3sconf/pdf/2023/28/e3sconf_icmed-icmpc2023_01048.pdf

[5] Joaquín Amat Rodrigo and Javier Escobar Ortiz, *Forecasting Energy Demand with Machine Learning*, 2021. https://cienciadedatos.net/documentos/py29-forecasting-electricity-power-demand-python

[6] Avijit Paul Piyal, Naimul Hasan Adnan, Md. Minhazur Rahman Shuvo, et al., *Energy Demand Forecasting Using Machine Learning: Perspective Bangladesh*, 2023 IEEE Region 10 Symposium (TENSYMP). https://ieeexplore.ieee.org/document/10087679

[7] G. Vijendar Reddy, Lakshmi Jaswitha Aitha, Ch. Poojitha, et al., *Electricity Consumption Prediction Using Machine Learning*, E3S Web of Conferences 391, 01048 (2023). https://www.e3s-conferences.org/articles/e3sconf/pdf/2023/28/e3sconf_icmed-icmpc2023_01048.pdf

[8] Booshra Nazifa Mahmud, Zannatul Ferdoush, Lamia Tasnim Mim, *Modelling and Forecasting Energy Demand of Bangladesh Using AI-Based Algorithms*, BRAC University, Undergraduate Thesis, 2020. https://dspace.bracu.ac.bd/xmlui/bitstream/handle/10361/12353/15301020,%2015301068,%2015301052_CSE.pdf?sequence=1

[9] IBM. (2024). *What is data leakage in machine learning?* IBM Think. https://www.ibm.com/think/topics/data-leakage-machine-learning

[10] GeeksforGeeks. (2024, September 19). *What is Lag in Time Series Forecasting*. Retrieved from https://www.geeksforgeeks.org/what-is-lag-in-time-series-forecasting/

[11] tsfresh Documentation. (n.d.). *Rolling / Time series forecasting*. Retrieved from https://tsfresh.readthedocs.io/en/latest/text/forecasting.html

[12] Raheja, S. (2025, January 8). *Train-Test-Validation Split in 2025*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/

[13] Scikit-learn documentation:
MinMaxScaler — sklearn.preprocessing.MinMaxScaler
Available at:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[14] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer. https://link.springer.com/book/10.1007/978-1-0716-1418-1

[15] GeeksforGeeks: Linear Regression in Machine Learning.
https://www.geeksforgeeks.org/ml-linear-regression/

[16] GeeksforGeeks: "Python | Decision Tree Regression using sklearn" *(Last Updated: Jan 29, 2025)*: https://www.geeksforgeeks.org/random-forest-regression-in-python/

[17] GeeksforGeeks, "Random Forest Regression in Python" (2025)
https://www.geeksforgeeks.org/random-forest-regression-in-python/

[18] GeeksforGeeks, "Implementing the AdaBoost Algorithm From Scratch" (2025)
https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/

[19] GeeksforGeeks, "XGBoost for Regression" (2023)
https://www.geeksforgeeks.org/xgboost-for-regression/

[20] GeeksforGeeks, “SVR using Linear and Non-Linear Kernels in Scikit-Learn” (Last Updated: 30 Jan, 2023)
https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/

[21] GeeksforGeeks, *“K-Nearest Neighbors (KNN) Regression with Scikit-Learn”* (17 Jun, 2024)
https://www.geeksforgeeks.org/k-nearest-neighbors-knn-regression-with-scikit-learn/

[22] Mobarak Inuwa, *“Non-Generalization and Generalization of ML Models”*, GeeksforGeeks, 08 Nov 2022
https://www.analyticsvidhya.com/blog/2022/10/non-generalization-and-generalization-of-machine-learning-models/

[23] GeeksforGeeks, "Voting Regressor," *GeeksforGeeks*, Oct. 25, 2023. [Online]. Available: https://www.geeksforgeeks.org/voting-regressor/

[24] GeeksforGeeks, "Stacking in Machine Learning," *GeeksforGeeks*, May 20, 2019. [Online]. Available: https://www.geeksforgeeks.org/stacking-in-machine-learning/

[25] “Explainable Artificial Intelligence (XAI),” *GeeksforGeeks,* April 15, 2025.
https://www.geeksforgeeks.org/explainable-artificial-intelligencexai/

Work Distribution

Task	Group Members	Remarks
Data Collection (Oct.-Dec. 2023), (Sept-Dec. 2022), (Jan-April 2022), (Oct.-Dec. 2021) = **14 months**	**Al Imran** **Momtahina Quyyum** **Shadman Shakib Dip**	Data collected collaboratively by three members.
Data Collection (Remaining months) - **34 months**	**Hasibul Hasan**	Collected remaining data individually (**34 months**).
Data Cleaning and Preprocessing	**Hasibul Hasan**	Performed data cleaning, handling missing values, feature engineering.
Model Development & Implementation	**Hasibul Hasan**	Implemented ML models and evaluated results.
Result Analysis & Visualization	**Hasibul Hasan**	Analyzed model results with graphs and interpretation.
Report Writing	**Hasibul Hasan**	Drafted and finalized the entire project report.