

Predictive Analytics for Energy Planning:

A Data-Driven Study on India's Energy Requirements and Availability

MACHINE LEARNING EDA REPORT

Submitted to

DR. PIYUSH CHAUHAN

Associate Professor

*Department of Computer Science &
Engineering*

*Symbiosis Institute of Technology, Nagpur
Campus*

Submitted by

Tanay Yaduka

VII SEM

PRN: 22070521086

*Department of Computer Science &
Engineering*

*Symbiosis Institute of Technology, Nagpur
Campus*

Course Name: Machine Learning

Course Code: T7529



॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS
INSTITUTE OF TECHNOLOGY, NAGPUR

Contents

1	Project Definition and Data Understanding	4
1.1	Problem Statement	4
1.2	Data Sources	4
1.3	Data Dictionary	4
2	Data Collection and Integration	6
2.1	Data Source and Collection	6
2.2	Data Provenance	6
2.3	Integration Methodology	6
2.4	Initial Validation	6
3	Data Cleaning and Preparation	7
3.1	Check Shape, Data Types, and Null Values	7
3.2	Normalize column names (strip/lower/replace non-alnum with underscore) .	8
3.3	Trim Whitespace and Standardize Text Columns:	9
3.4	Simple outlier detection (IQR) for numeric columns	10
3.5	Final Checks Save Cleaned File	11
4	Exploratory Data Analysis	13
4.1	Descriptive Statistics	13
4.2	Univariate Analysis	15
4.3	Bivariate Analysis	17
4.4	Correlation Analysis	19
4.5	Time Series Trends	21
4.5.1	Energy Requirement vs Availability by Month	21
4.5.2	Energy Deficit Trend by Region	23
4.5.3	Top 10 States by Average Energy Deficit	24
4.5.4	Average Energy Deficit by Month (All Regions)	25
4.5.5	Monthly Energy Deficit by Quarter	26
4.6	Average Energy Deficit: State vs Month	27
4.7	Average Energy Deficit: Region vs Month	29
4.8	Pairwise Relationships by Region	30
4.9	Test for Statistical Properties	32
4.9.1	Normality Tests	32
5	Machine Learning Algorithms Overview	35
5.1	Prepare dataset	35
5.2	Preprocessing for Base and Seasonality Models	36
6	Train and Compare Multiple Models	38
6.1	KNN Classifier (Base vs Seasonality)	38
6.2	Naive Bayes (Base vs Seasonality)	41
6.3	Logistic Regression (Base vs Seasonality)	43
6.4	Linear Regression (Base vs Seasonality)	46

7	Machine Learning Model Evaluation Summary	48
8	Decision Analysis	49
9	Research Work	50
9.1	Evaluation of Research on Energy Efficiency in Energy-Intensive Manufacturing Companies	50
9.2	A Study on Energy Consumption, Energy Saving, and Effectiveness of Alternate Energy Sources in India's Domestic Sector	50
9.3	Renewable Energy in India: Current Status and Future Potentials	50
9.4	A Study on Energy Consumption, Energy Saving, and Effectiveness of Alternate Energy Sources in the Domestic Sector of India	51
9.5	Renewable Energy in India: Current Status and Future Potentials	51
10	Conclusion and Recommendations	52
10.1	Key Takeaways	52
10.2	Business Recommendations	52
10.3	Future Work	52

1 Project Definition and Data Understanding

1.1 Problem Statement

In the rapidly evolving landscape of energy consumption and production in India, policy-makers and energy providers face challenges in balancing energy requirements with actual availability across different states. A mismatch between required and available energy can lead to power shortages, economic losses, and reduced quality of life. Therefore, having a reliable method to analyze historical energy data and forecast future energy needs is critical.

This project aims to use statistical and machine learning techniques to analyze patterns in energy requirement and availability across Indian states and develop predictive models to support data-driven decision-making.

1.2 Data Sources

The primary dataset used in this project was obtained from the official Government of India data portal:

- **Source:** data.gov.in — Open Government Data (OGD) Platform India
- **Dataset Name:** Energy Requirement and Availability
- **Format:** CSV
- **Coverage:** State-wise annual energy requirement and availability data
- **Access Date:** October 26, 2025

1.3 Data Dictionary

Field Name	Description	Data Type	Example Values
id	Unique identifier for each record	Integer	1, 2, 3
state_code	Official state code (abbreviated or numeric)	Categorical / Integer	27, 09, 29
state	Abbreviated name of the state or UT	Categorical (String)	MH, UP, KA
name	Full name of the state or UT	Categorical (String)	Maharashtra, Uttar Pradesh, Karnataka
energy_requirement	Total energy required in a given year (in MU)	Numerical (Float)	130450.0, 98500.5
energy_availability	Total energy available in a given year (in MU)	Numerical (Float)	128900.0, 96000.0

Table 1: Data Dictionary for Energy Dataset from India Gov Portal

Delimiter: , 								
	Region	State	is_union_territory	Month	Quarter	Energy Requirement MU	Energy Availability MU	Energy_Deficit
1	North	Chandigarh	1	Mar	Q1	101.6	101.6	0
2	North	Delhi	1	Mar	Q1	2130.478	2130.279	0.199
3	North	Haryana	0	Mar	Q1	3911.822	3909.16	2.662
4	North	Himachal Pradesh	0	Mar	Q1	1025.63	1023.16	2.47
5	North	UTs of J&K and Ladakh	1	Mar	Q1	1725.61	1717.93	7.68
6	North	Punjab	0	Mar	Q1	4275.01	4259.19	15.82
7	North	Rajasthan	0	Mar	Q1	7541.57	7509.03	32.54
8	North	Uttar Pradesh	0	Mar	Q1	9619.29	9613.89	5.4
9	North	Uttarakhand	0	Mar	Q1	1161.761	1149.01	12.751
10	Central	Chhattisgarh	0	Mar	Q1	3522.615219	3519.613719	3.0015
11	West	Gujarat	0	Mar	Q1	11336.89609	11336.89609	0
12	Central	Madhya Pradesh	0	Mar	Q1	7524.602891	7496.421383	28.18150782
13	West	Maharashtra	0	Mar	Q1	17010.05555	17009.96633	0.08922
14	West	Haveli and Daman & Diu	1	Mar	Q1	791.7707	791.7707	0
15	West	Goa	0	Mar	Q1	405.46282	405.46282	0
16	South	Andhra Pradesh	0	Mar	Q1	6711.89014	6711.89014	0
17	South	Telangana	0	Mar	Q1	8909.39502	8909.39502	0
18	South	Karnataka	0	Mar	Q1	8602.621995	8602.621995	0
19	South	Kerala	0	Mar	Q1	2713.099073	2713.099073	0
20	South	Tamil Nadu	0	Mar	Q1	10799.8363	10799.8363	0
21	South	Puducherry	1	Mar	Q1	263.8251642	263.8251642	0
22	South	Lakshadweep	1	Mar	Q1	6.01565	6.01565	0
23	East	Bihar	0	Mar	Q1	2882.105319	2863.953819	18.1515
24	East	Jharkhand	0	Mar	Q1	1106.484301	1034.630301	71.854
25	East	Odisha	0	Mar	Q1	3441.362624	3441.362624	0
26	East	West Bengal	0	Mar	Q1	4982.247482	4977.483482	4.764
27	Northeast	Sikkim	0	Mar	Q1	50.49075133	50.49075133	0
28	South	Andaman Nicobar	1	Mar	Q1	31.82	31.37	0.45
29	Northeast	Arunachal Pradesh	0	Mar	Q1	86.89744396	83.19240168	3.705042279
30	Northeast	Assam	0	Mar	Q1	843.6235207	843.6235207	0

Figure 1: Preview of the Energy Dataset

2 Data Collection and Integration

2.1 Data Source and Collection

The dataset used in this project was sourced from the Open Government Data (OGD) Platform India (<https://data.gov.in>). It provides annual statistics on energy requirement and availability for various states and union territories of India. The data was obtained in CSV format and is publicly available for academic and analytical use.

2.2 Data Provenance

- **Source Authority:** Ministry of Power, Government of India
- **Dataset Format:** CSV (Comma-Separated Values)
- **Date of Access:** October 26, 2025
- **License:** Open Government License – India

2.3 Integration Methodology

The dataset was integrated into the analysis pipeline using Python. It was loaded into a DataFrame using the Pandas library:

```
1 import pandas as pd
2 df = pd.read_csv("energy-requirement-and-availability.csv")
```

As the dataset was already well-structured and complete, no merging with additional datasets was necessary. All fields were verified for consistency in naming and data type.

2.4 Initial Validation

The dataset underwent an initial validation process:

- All column headers were correctly named and interpretable.
- No duplicate records or missing values were found at this stage.
- The schema followed the expected structure with unique state-year entries.

3 Data Cleaning and Preparation

3.1 Check Shape, Data Types, and Null Values

To begin the data cleaning process, the structure and integrity of the dataset were examined. The dataset consists of a total of **1534 rows and 6 columns**, providing an initial understanding of its scale.

Next, the data types of each column were checked to ensure consistency with expected formats. Numerical fields such as `energy_requirement` and `energy_availability` were correctly stored as floating-point values. Identifier fields like `id` and `state_code` appeared as integers. Textual fields such as `state` and `name` were stored as object types (strings), and may be converted to categorical types to optimize memory usage and prepare for machine learning tasks.

A null value check was also conducted across all columns. The dataset did not contain any missing values, indicating it is complete and does not require imputation or deletion of records. This makes the data ready for further preprocessing and analysis.

```
1 # Dataset shape
2 print("Dataset shape:", df.shape)
3
4 # Data types and nulls
5 print("\nData types and missing values:")
6 print(df.info())
7
8 # Summary of null values
9 print("\nMissing values in each column:")
10 print(df.isnull().sum())
```

```
Dataset shape: (1534, 6)
```

```
Data types and missing values:
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 350 entries, 0 to 349
```

```
Data columns (total 8 columns):
```

#	Column	Non-Null	Count	Dtype
0	Region	350	non-null	object
1	State	350	non-null	object
2	is_union_territory	350	non-null	int64
3	Month	350	non-null	object
4	Quarter	350	non-null	object
5	Energy Requirement MU	350	non-null	float64
6	Energy Availability MU	350	non-null	float64
7	Energy Deficit	350	non-null	float64

```
dtypes: float64(3), int64(1), object(4)
```

```
memory usage: 22.0+ KB
```

```
None
```

```
Missing values in each column:
```

```
Region          0
```

```
State          0
is_union_territory 0
Month          0
Quarter        0
Energy Requirement MU 0
Energy Availability MU 0
Energy_Deficit 0
dtype: int64
```

3.2 Normalize column names (strip/lower/replace non-alnum with underscore)

It means cleaning up your dataset's column names to make them consistent and easy to use in code.

- **strip()** → Removes extra spaces at the start or end of the column name.
- **lower()** → Converts all letters to lowercase, e.g., `Customer Name` → `customer name`.
- **replace non-alnum with underscore** → Replaces any character that is not a letter or number (like spaces, `-`, `#`, etc.) with an underscore, e.g., `customer name` → `customer_name`.

Importance:

Normalizing column names is an important step in data cleaning because it:

- Ensures **consistency** across all column names, avoiding confusion or errors.
- Makes it easier to **access columns programmatically** without syntax issues (e.g., spaces or special characters).
- Helps maintain **compatibility** with different tools, libraries, and programming languages.
- Improves **readability and professionalism** of the dataset.

Example: " Customer Name# " → "customer_name"

```
1 df.columns = (df.columns
2               .astype(str)
3               .str.strip()
4               .str.lower()
5               .str.replace(r'^0-9a-z|+', '_', regex=True)
6               .str.strip('_')
7               )
8
9 print("\nColumns after normalization:", list(df.columns))
```



```
Columns after normalization: ['region', 'state', 'is_union_territory', 'month',
                             'quarter', 'energy_requirement_mu', 'energy_availability_mu', 'energy_deficit']
```

Explanation:

The code normalizes all column names in the DataFrame `df` by:

- Converting names to strings using `astype(str)`
- Removing extra spaces with `str.strip()`
- Changing all letters to lowercase using `str.lower()`
- Replacing non-alphanumeric characters with underscores using regex
- Removing leading and trailing underscores

Importance:

This process improves readability, avoids syntax errors, and ensures smooth data manipulation in Python.

3.3 Trim Whitespace and Standardize Text Columns:

This step involves cleaning text-based columns in a dataset by removing unnecessary spaces and ensuring consistency in text formatting.

- **Trimming whitespace:** Removes leading and trailing spaces using functions like `str.strip()`, ensuring that text values are clean and uniform.
- **Standardizing text:** Converts all text to a common case (usually lowercase) and may replace inconsistent characters or formats (e.g., converting “Yes”, “YES”, and “yes” to a single standard form “yes”).

Importance:

- Prevents mismatches and duplication caused by inconsistent text entries.
- Ensures accurate grouping, filtering, and merging of data.
- Improves the quality and reliability of data analysis and machine learning models.

Example:

" Delhi " → "delhi"

```
1 text_cols = df.select_dtypes(include=['object']).columns.tolist()
2 for c in text_cols:
3     df[c] = df[c].astype(str).str.strip() # keep as str for parsing step
4
5 # Title-case region/state (but keep 'month' / 'quarter' as-is)
6 if 'region' in df.columns:
7     df['region'] = df['region'].str.title()
8 if 'state' in df.columns:
9     df['state'] = df['state'].str.title()
```

Explanation:

The given code is used to clean and standardize text columns in a DataFrame `df`.

1. **Select text columns:**

```
text_cols = df.select_dtypes(include=['object']).columns.tolist()
```

This line identifies all columns in the DataFrame that contain text (`object` type) and stores their names in the list `text_cols`.

2. **Trim whitespace:**

```
df[c] = df[c].astype(str).str.strip()
```

For each text column, it converts values to strings and removes leading or trailing spaces to ensure data consistency.

3. **Standardize capitalization:**

The next lines check if columns named `'region'` or `'state'` exist. If they do, their text values are converted to title case using `str.title()`, so that:

`"delhi" → "Delhi"`

Columns like `'month'` and `'quarter'` are intentionally left unchanged.

Purpose:

This process ensures that text data is clean, consistent, and properly formatted, which helps avoid errors during data analysis, grouping, or merging.

3.4 Simple outlier detection (IQR) for numeric columns

The Interquartile Range (IQR) method is used to identify outliers in numeric data. It calculates the first quartile ($Q1$) and third quartile ($Q3$), and the IQR as $(Q3 - Q1)$. Any value below $(Q1 - 1.5 \times IQR)$ or above $(Q3 + 1.5 \times IQR)$ is considered an outlier. This method helps detect extreme or unusual values, improving data quality and ensuring that analysis or models are not skewed by anomalies.

```
1 outlier_report = {}
2 for col in numeric_cols:
3     q1 = df[col].quantile(0.25)
4     q3 = df[col].quantile(0.75)
5     iqr = q3 - q1
6     lower = q1 - 1.5*iqr
7     upper = q3 + 1.5*iqr
8     mask = (df[col] < lower) | (df[col] > upper)
9     outlier_count = mask.sum()
10    outlier_report[col] = {'count': int(outlier_count), 'lower': float(lower),
11                           'upper': float(upper)}
11 print("\nOutlier report (IQR method):")
12 for k,v in outlier_report.items():
13     print(f" - {k}: {v['count']} outliers (outside [{v['lower']:.3f}, {v['upper']:.3f}])")
```

```

Outlier report (IQR method):
- energy_requirement_mu: 7 outliers (outside [-9580.443, 16447.334])
- energy_availability_mu: 7 outliers (outside [-9614.250, 16459.750])
- energy_deficit: 58 outliers (outside [-9.977, 16.628])

```

Explanation:

Using the Inter-quartile Range (IQR) method, outliers were detected in the numeric columns of the dataset. The lower and upper bounds were calculated as $Q1 - 1.5 \times IQR$ and $Q3 + 1.5 \times IQR$, respectively, and values outside this range were counted. The resulting report is:

- **energy_requirement_mu:** 7 outliers (outside [-9580.443, 16447.334])
- **energy_availability_mu:** 7 outliers (outside [-9614.250, 16459.750])
- **energy_deficit:** 58 outliers (outside [-9.977, 16.628])

This indicates that **energy_deficit** has a relatively larger number of extreme values, which may require further investigation or handling before analysis.

3.5 Final Checks Save Cleaned File

```

1 print("\nFinal shape:", df.shape)
2 display(df.head())

```

	region	state	is_union_territory	month	quarter	energy_requirement_mu	energy_availability_mu	energy_deficit
0	North	Chandigarh	True	Mar	Q1	101.600	101.600	0.000
1	North	Delhi	True	Mar	Q1	2130.478	2130.279	0.199
2	North	Haryana	False	Mar	Q1	3911.822	3909.160	2.662
3	North	Himachal Pradesh	False	Mar	Q1	1025.630	1023.160	2.470
4	North	Uts Of J&K And Ladakh	True	Mar	Q1	1725.610	1717.930	7.680

Figure 2: Columns in data-frame

```

1 cleaned_path = "cleaned_dataset.csv"
2 df.to_csv(cleaned_path, index=False)
3 print(f"\nCleaned dataset saved to: {cleaned_path}")

```

```
Cleaned dataset saved to: cleaned_dataset.csv
```

```

1 print("\nData types summary:")
2 print(df.dtypes)

```

```

Data types summary:
region          object
state           object
is_union_territory  bool
month           object
quarter         object

```

```
energy_requirement_mu    float64
energy_availability_mu    float64
energy_deficit            float64
dtype: object
```

4 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a critical step in the data science process that involves visually and statistically examining the dataset to uncover patterns, detect anomalies, identify relationships between variables, and test hypotheses. The primary objective of EDA is to better understand the structure, distribution, and nature of the data before applying statistical modeling or machine learning techniques.

EDA helps analysts gain insights into the dataset by using descriptive statistics, visualizations, and correlation analysis. This phase plays a key role in decision-making for data preprocessing, feature selection, and model design by revealing hidden trends and potential data issues. It ensures that the subsequent analysis is grounded in a clear and accurate understanding of the data.

4.1 Descriptive Statistics

Descriptive statistics provide a fundamental summary of the dataset and help in understanding the central tendency, spread, and shape of the distribution of numerical variables. In this project, descriptive metrics such as mean, standard deviation, minimum, maximum, and quartiles were calculated for the `energy_requirement_mu`, `energy_availability_mu`, and `energy_deficit` columns:

```
1 # Load dataset
2 df = pd.read_csv("cleaned_dataset.csv")
3 # Data types
4 print(df.dtypes)
5
6 # Missing values
7 print(df.isnull().sum())
8
9 # Summary statistics
10 print(df.describe(include="all"))
```

This statistical overview is essential during EDA as it highlights patterns, detects inconsistencies, and guides data transformation or modeling decisions.

```

region      object
state       object
is_union_territory  bool
month       object
quarter     object
energy_requirement_mu  float64
energy_availability_mu  float64
energy_deficit  float64
dtype: object
region      0
state       0
is_union_territory  0
month       0
quarter     0
energy_requirement_mu  0
energy_availability_mu  0
energy_deficit  0
dtype: int64

```

	region	state	is_union_territory	month	quarter	\
count	350	350	350	350	350	
unique	6	35		2	10	4
top	North	Chandigarh	False	Mar	Q2	
freq	90	10		280	35	105
mean	NaN	NaN		NaN	NaN	NaN
std	NaN	NaN		NaN	NaN	NaN
min	NaN	NaN		NaN	NaN	NaN
25%	NaN	NaN		NaN	NaN	NaN
50%	NaN	NaN		NaN	NaN	NaN
75%	NaN	NaN		NaN	NaN	NaN
max	NaN	NaN		NaN	NaN	NaN

	energy_requirement_mu	energy_availability_mu	energy_deficit
count	350.000000	350.000000	350.000000
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	3795.145090	3785.631188	9.513902
std	4434.033361	4429.376747	21.834652
min	5.000000	5.000000	0.000000
25%	179.973259	163.500000	0.000000
50%	1596.000000	1576.500000	0.000000
75%	6686.917605	6682.000000	6.651344
max	18056.000000	18051.000000	195.000000

Figure 3: Columns in data-frame

The descriptive statistics provide several valuable insights into the energy-related data. The mean values for the three numerical variables — `energy_requirement_mu`, `energy_availability_mu`, and `energy_deficit` — are 3795.15, 3785.63, and 9.51, respectively. The standard deviations indicate considerable variability in `energy_requirement_mu` and `energy_availability_mu` (both around 4430), while `energy_deficit` shows lower dispersion (21.83), suggesting that for most states, the availability closely follows the requirement.

The median values (50% quantile) for `energy_requirement_mu` and `energy_availability_mu` are 1596.00 and 1576.50, respectively, indicating that half of the observations lie below the mean. The median of `energy_deficit` is 0.0, suggesting that most states experience little to no deficit.

Furthermore, the range between the minimum and maximum values highlights the presence of both surplus and deficit conditions: `energy_requirement_mu` and `energy_availability_mu` span from 5.0 to over 18000, while `energy_deficit` ranges from 0.0 to 195.0. The relatively small interquartile range (IQR) of `energy_deficit` (0.0 – 6.65) indicates that extreme deficits are rare and most values are clustered near zero, reflecting a generally balanced energy supply-demand scenario.

These observations provide a strong foundation for deeper visual and statistical analysis in subsequent steps.

Visualization

Visualizations are powerful tools in Exploratory Data Analysis (EDA) as they allow us to interpret data patterns, relationships, and distributions quickly and intuitively. Graphical representations such as histograms, box plots, scatter plots, and heatmaps make it easier to detect outliers, skewness, clusters, and correlations that may not be immediately obvious through statistical summaries alone.

By transforming numerical values into visual insights, we can identify trends, seasonal variations, and anomalies that influence decision-making and model building. Effective visualization supports clearer communication of findings and ensures a deeper understanding of the data structure and behavior.

4.2 Univariate Analysis

Univariate analysis is the examination of a single variable at a time to understand its distribution, central tendency, and variability. It is an important step in exploratory data analysis (EDA) because it helps identify patterns, detect anomalies, and summarize key characteristics of the data. For numeric variables, common techniques include histograms, boxplots, and descriptive statistics (mean, median, standard deviation), which reveal the spread and presence of outliers. For categorical variables, frequency counts and bar plots are used to assess the distribution of categories. Conducting univariate analysis provides a foundational understanding of each variable individually, which is essential before exploring relationships between variables in multivariate analysis.

```
1 # Histograms of numeric columns
2 df[['energy_requirement_mu', 'energy_availability_mu', 'energy_deficit']].hist(
    figsize=(12,6), bins=30)
3 plt.suptitle("Distribution of Numeric Features", fontsize=14)
4 plt.show()
5
6 # Boxplots for outliers
7 plt.figure(figsize=(12,6))
8 sns.boxplot(data=df[['energy_requirement_mu', 'energy_availability_mu',
    'energy_deficit']])
9 plt.title("Boxplots of Numeric Features")
10 plt.show()
11
12 # Count plots for categorical features
13 plt.figure(figsize=(12,5))
14 sns.countplot(data=df, x="region", order=df['region'].value_counts().index)
15 plt.xticks(rotation=90)
16 plt.title("Region-wise Records")
17 plt.show()
18
19 plt.figure(figsize=(8,5))
20 sns.countplot(data=df, x="quarter")
21 plt.title("Quarter Distribution")
22 plt.show()
```

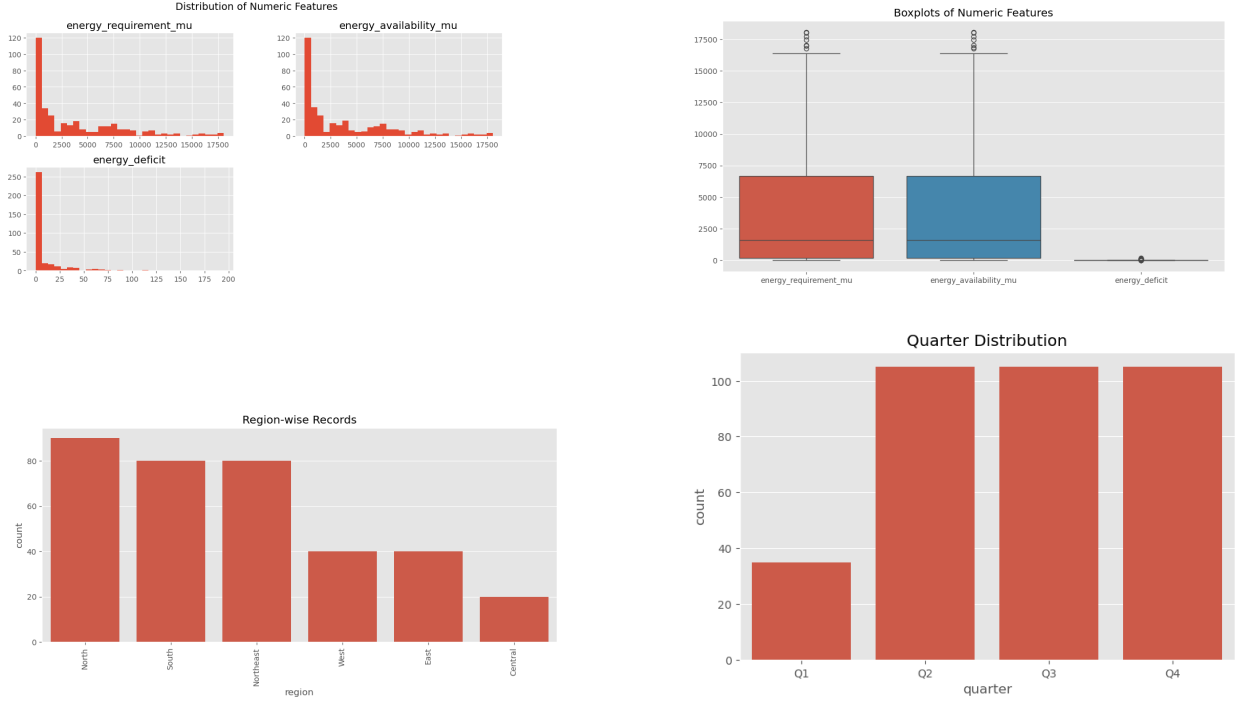


Figure 4: Uni-variate Analysis

Categorical Feature Distribution

- **Region-wise Records:**

- The dataset is highly skewed towards the **North** (≈ 90 records), **South**, and **Northeast** (both ≈ 80 records).
- **Central** has the lowest count (≈ 20 records).

- **Quarter Distribution:**

- The data is heavily concentrated in **Q2**, **Q3**, and **Q4** (each ≈ 105 records).
- **Q1** is significantly underrepresented (≈ 35 records).

Numeric Feature Analysis

The features `energy_requirement_mu`, `energy_availability_mu`, and `energy_deficit` are all characterized by **strong positive skewness** (right-skewed).

- **`energy_requirement_mu` & `energy_availability_mu`:**

- The majority of values are near **zero**, as indicated by a very low median.
- Both features show a large **Interquartile Range** (IQR) up to $\approx 7,000$.
- Numerous extreme **outliers** are present, extending up to $\approx 18,000$.

- **`energy_deficit`:**

- Values are highly clustered around **zero**; the median and IQR are near **0**.
- Shows a few small **outliers** up to ≈ 200 , but on a much smaller scale than the other two features.

4.3 Bivariate Analysis

Bivariate analysis examines the relationship between two variables to identify patterns, correlations, or potential dependencies. It is an essential part of exploratory data analysis (EDA) because it helps uncover insights that are not visible when analyzing variables individually. For numeric-numeric pairs, scatter plots, correlation coefficients, and covariance are commonly used to assess linear or non-linear relationships. For numeric-categorical pairs, boxplots or violin plots can reveal differences in distributions across categories. For categorical-categorical pairs, cross-tabulations and stacked bar charts help identify associations. Conducting bivariate analysis provides a deeper understanding of how variables interact, which is crucial for feature selection, predictive modeling, and hypothesis testing.

```

1 # Requirement vs Availability
2 plt.figure(figsize=(8,6))
3 sns.scatterplot(data=df, x="energy_requirement_mu", y="energy_availability_mu",
4                 , hue="region", alpha=0.7)
5 plt.title("Energy Requirement vs Availability")
6 plt.show()
7
8 # Avg Deficit by Region
9 plt.figure(figsize=(12,6))
10 sns.barplot(data=df, x="region", y="energy_deficit", estimator=np.mean, order=
11             df.groupby('region')['energy_deficit'].mean().sort_values(ascending=False)
12             .index)
13 plt.xticks(rotation=90)
14 plt.title("Average Energy Deficit by Region")
15 plt.show()
16
17 # Deficit by Quarter
18 plt.figure(figsize=(8,5))
19 sns.boxplot(data=df, x="quarter", y="energy_deficit")
20 plt.title("Energy Deficit by Quarter")
21 plt.show()

```

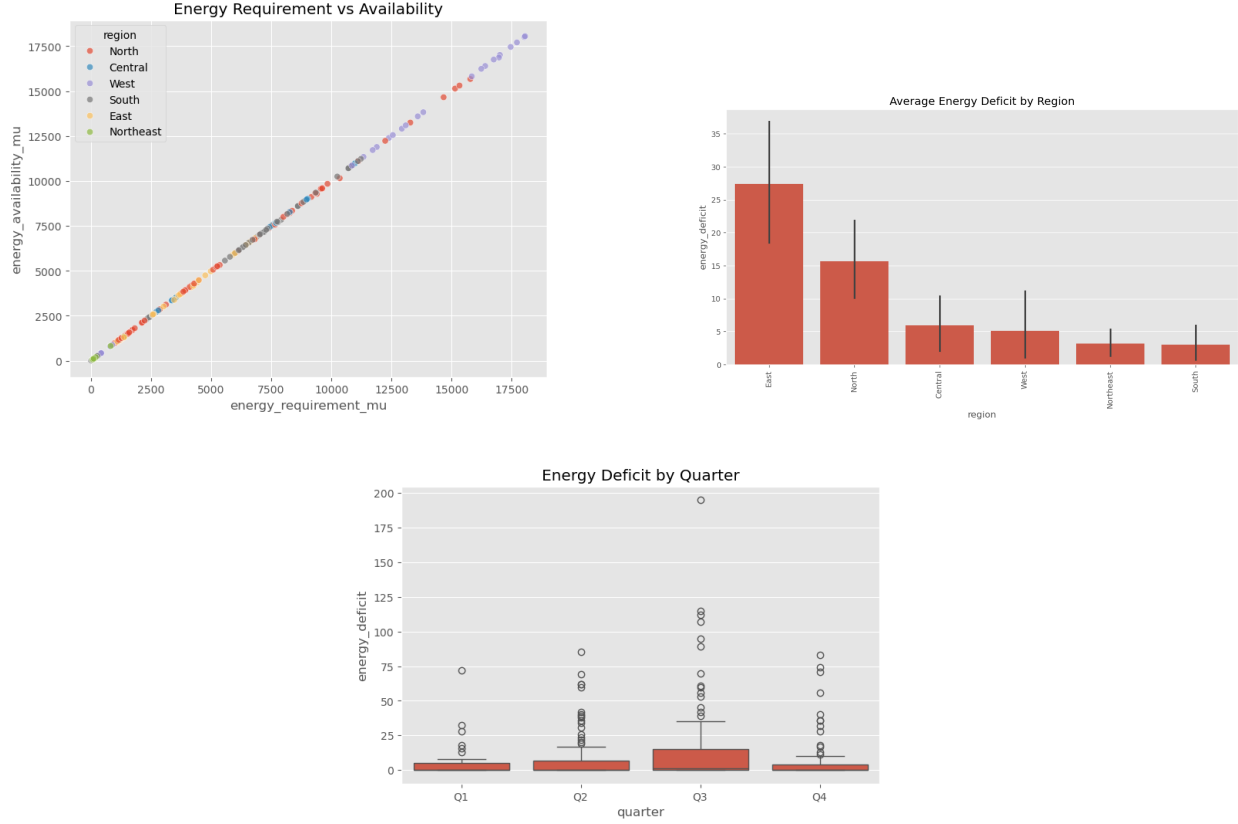


Figure 5: Uni-variate Analysis

Energy Requirement vs Availability (Scatter Plot)

- **Strong Positive Correlation:** The scatter plot shows a nearly **perfect linear relationship** between energy_requirement_mu and energy_availability_mu. This indicates that as the required energy increases, the available energy increases at a highly proportional rate.
- **Near Zero Deficit:** Since the points fall extremely close to the line $y = x$, this implies that for most data points, energy_availability_mu is almost exactly equal to energy_requirement_mu, resulting in a minimal energy_deficit.
- **Region Distribution:** Data points from all regions (North, Central, West, South, East, Northeast) are distributed uniformly along the correlation line, suggesting this high correlation holds true across all regions.

Average Energy Deficit by Region (Bar Chart with Error Bars)

- **Highest Average Deficit:** The **East** region has the highest average energy deficit (≈ 28), followed by the **North** (≈ 16).
- **Lowest Average Deficit:** The **Northeast** and **South** regions show the lowest average deficit (≈ 3).

- **Variability (Error Bars):** The error bars (representing standard deviation or confidence interval) are substantial, particularly for the **East** and **North** regions. This indicates high variability (or wide data spread) in the energy deficit within these regions. The large error bar for the East suggests the average is driven by a few very high deficit values.

Energy Deficit by Quarter (Box Plot)

- **Median Deficit:** For all quarters (**Q1**, **Q2**, **Q3**, **Q4**), the median energy_deficit (the line inside the box) is very close to **zero**.
- **Variability:** **Q3** shows the largest Interquartile Range (IQR) (≈ 0 to ≈ 15) and the highest maximum values/outliers.
- **Extreme Outliers:**
 - **Q3** exhibits the most extreme outlier, nearly reaching **200**.
 - **Q2** and **Q4** also have numerous outliers, but the maximum values are lower (≈ 115 for Q3; ≈ 85 for Q4).
 - **Q1** has the lowest overall spread and maximum outlier value (≈ 75).
- **Conclusion:** While the typical energy deficit is negligible across all quarters, the **third quarter (Q3)** experiences the largest and most extreme deficits.

4.4 Correlation Analysis

Correlation analysis measures the strength and direction of the relationship between two numeric variables. It is an important part of exploratory data analysis (EDA) because it helps identify which variables are positively or negatively associated, and which may be redundant. Common techniques include calculating the Pearson correlation coefficient for linear relationships, the Spearman rank correlation for monotonic relationships, and visualizing correlations using heatmaps or pair plots. Understanding correlations aids in feature selection, detecting multicollinearity, and guiding further statistical modeling or machine learning analysis.

```
1 plt.figure(figsize=(8,6))
2 sns.heatmap(df[['energy_requirement_mu', 'energy_availability_mu', '
   energy_deficit']].corr(), annot=True, cmap="coolwarm")
3 plt.title("Correlation Heatmap")
4 plt.show()
```

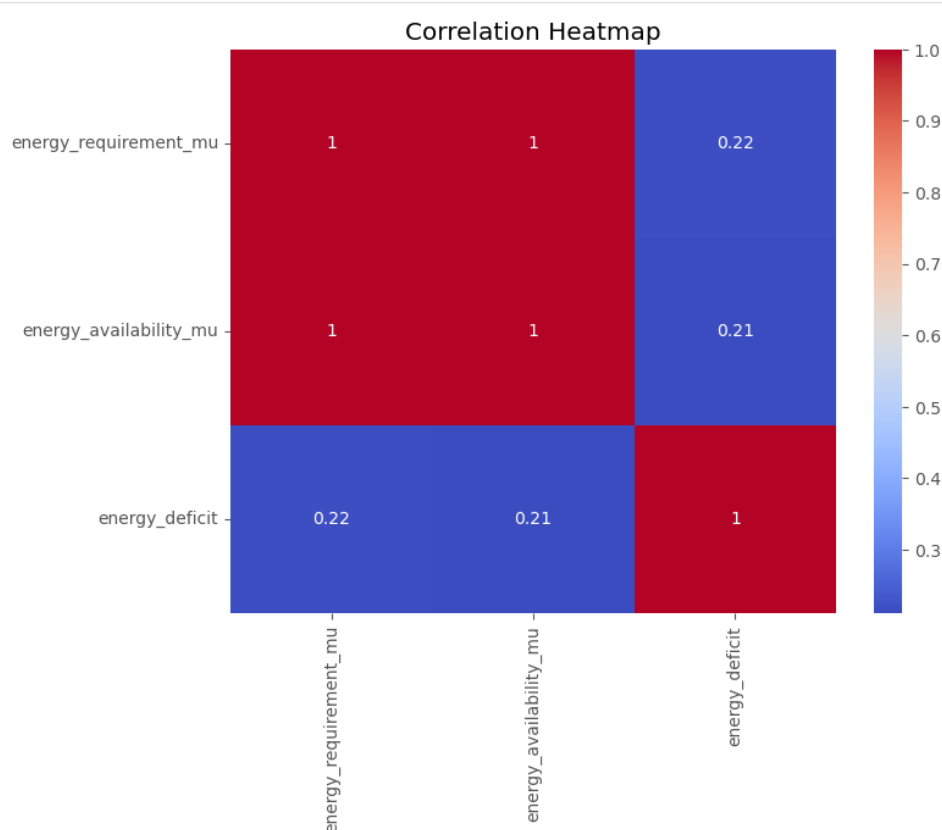


Figure 6: Boxplots for Energy Metrics

The heatmap displays the Pearson correlation coefficients among the three numeric features: `energy_requirement_mu`, `energy_availability_mu`, and `energy_deficit`.

- **Perfect Positive Correlation (1):**

- The correlation between `energy_requirement_mu` and `energy_availability_mu` is **1**.
- This confirms the observation from the scatter plot (previous analysis) that these two variables are **perfectly linearly related**. They move together in lockstep, meaning high requirement is always matched by high availability, and vice-versa.

- **Low Positive Correlation (0.21 to 0.22):**

- The correlation between `energy_deficit` and `energy_requirement_mu` is **0.22**.
- The correlation between `energy_deficit` and `energy_availability_mu` is **0.21**.
- These values indicate a **very weak positive correlation**. Although statistically significant, this relationship suggests that higher energy requirement/availability only slightly contributes to a larger energy deficit.
- Given the definition: $\text{Energy Deficit} = \text{Requirement} - \text{Availability}$, and the fact that $\text{Requirement} \approx \text{Availability}$, the deficit itself is mostly close to zero, explaining the near-zero correlation with its constituent parts. The slight positive value

is likely driven by the few extreme outliers where the small difference is magnified by the large magnitude of the requirement/availability values.

4.5 Time Series Trends

Time series trend analysis involves examining data points collected or recorded at regular time intervals to identify underlying patterns, trends, or seasonal effects. It is important in exploratory data analysis (EDA) because it helps understand how variables evolve over time, detect long-term increases or decreases, and identify periodic fluctuations. Visualizations such as line plots, rolling averages, and seasonal decomposition are commonly used to highlight trends, cycles, and anomalies. Analyzing time series trends provides valuable insights for forecasting, planning, and decision-making based on temporal dynamics.

4.5.1 Energy Requirement vs Availability by Month

```
1 month_order = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
2 df["month"] = pd.Categorical(df["month"], categories=month_order, ordered=True)
3
4 # -----
5 # 1. Energy Requirement vs Availability (by Month)
6 # -----
7 plt.figure(figsize=(12,6))
8 sns.lineplot(data=df, x="month", y="energy_requirement_mu", label="Requirement", marker="o")
9 sns.lineplot(data=df, x="month", y="energy_availability_mu", label="Availability", marker="o")
10 plt.title("Energy Requirement vs Availability by Month")
11 plt.ylabel("Energy (MU)")
12 plt.xlabel("Month")
13 plt.legend()
14 plt.show()
```

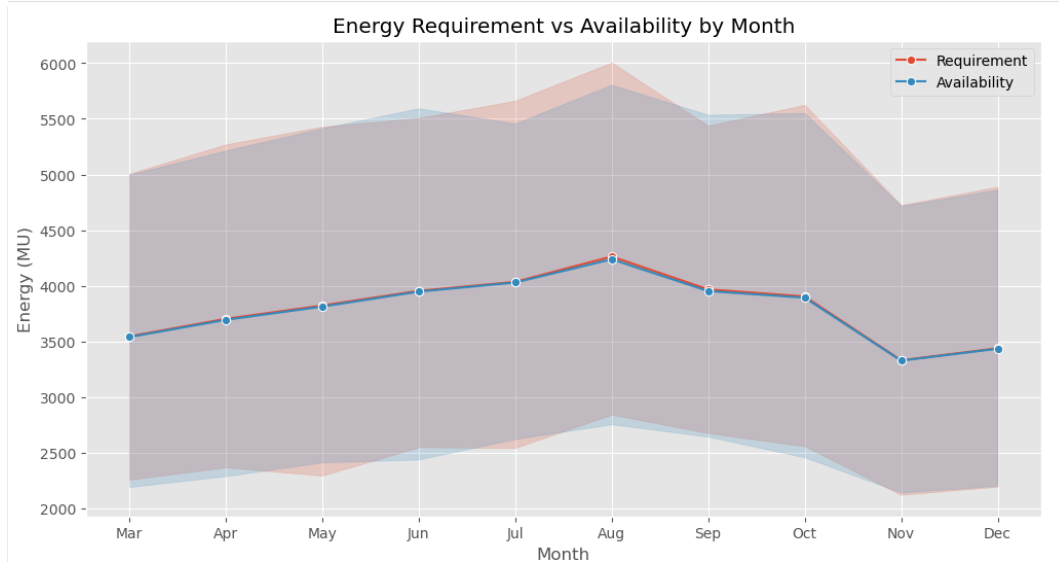


Figure 7: Energy Requirement vs Availability (by Month)

The line plot illustrates the mean monthly trends for Energy Requirement (MU) and Energy Availability (MU), with the shaded area likely representing the variability (e.g., standard deviation or range).

- **High Correlation/Near Equality:** The lines for Requirement (red) and Availability (blue) are almost perfectly superimposed throughout the year, confirming the high correlation observed previously. The **mean requirement is nearly identical to the mean availability** in every month.
- **Peak Energy Period (Mid-Year):**
 - Both energy requirement and availability **peak in August** at approximately **4,250 MU**.
 - They show a steady increase from March to August (March $\approx 3,550$ MU).
- **Trough Energy Period (Year-End):**
 - After the August peak, both metrics steadily **decline to their lowest point in November** ($\approx 3,350$ MU).
 - There is a slight recovery in December.
- **Variability (Shaded Area):**
 - The **variability** (spread between the upper and lower bounds of the shaded region) for both requirement and availability is **highest during the peak summer months** (July and August) and generally remains large throughout the year, indicating a significant difference between minimum and maximum recorded values for these months.
 - The variability **decreases noticeably in November**, which corresponds to the overall trough in energy values.

4.5.2 Energy Deficit Trend by Region

```
1 # Energy Deficit by Region (month trend)
2 plt.figure(figsize=(14,6))
3 sns.lineplot(data=df, x="month", y="energy_deficit", hue="region", marker="o")
4 plt.title("Energy Deficit Trend by Region")
5 plt.ylabel("Deficit (MU)")
6 plt.xlabel("Month")
7 plt.show()
```

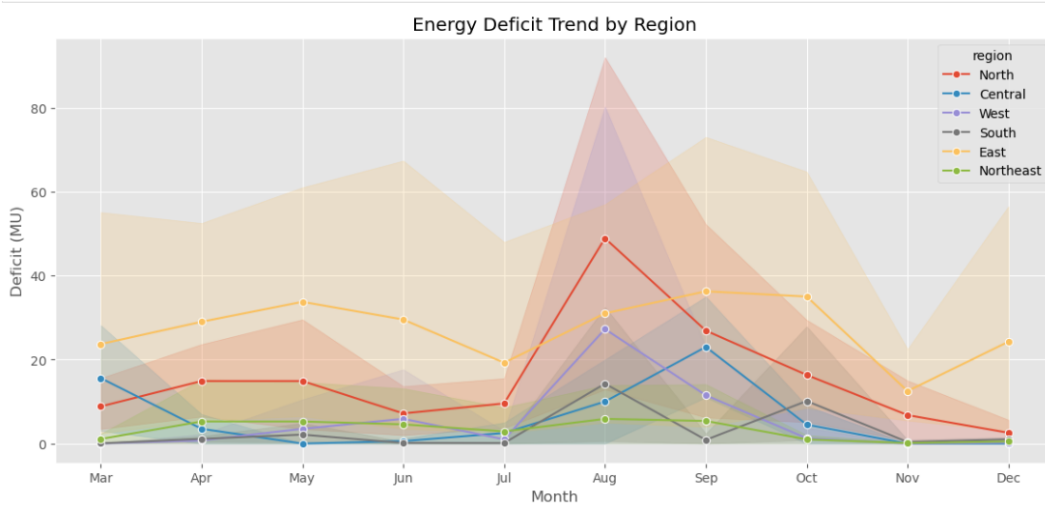


Figure 8: Energy Deficit by Region (month trend)

The line plot illustrates the monthly trend of the **mean energy deficit (MU)** for six different regions, with the shaded areas representing the variability (spread) of the deficit within each region.

- **Overall Deficit Peak:** The most significant finding is a sharp, distinct **peak in the energy deficit for all regions in August**. This confirms the earlier observation that the third quarter (which includes August) experiences the largest deficits.
- **Region-Specific Trends:**
 - **North (Red):** Shows the highest mean deficit peak, reaching nearly **50 MU** in August. Its overall variability (shaded area) is also very large, particularly around the August peak (≈ 90 MU).
 - **East (Orange):** Maintains the highest average deficit for most of the year (March-July, September-October). It peaks around **35 MU** in September, slightly after the overall August peak. Its variability is consistently high.
 - **West (Purple):** Exhibits a very sharp, extreme variability peak in August (≈ 95 MU), but its mean deficit line remains relatively lower than the North and East.
 - **Northeast (Green) & South (Black):** Consistently show the **lowest mean deficits** throughout the year, typically staying close to or below **5 MU**. Their variability is also the lowest.

- **Seasonality:**

- The deficit tends to be higher during the **summer months** (May to October).
- The deficit **dips significantly in November** for all regions, then shows a slight rise again in December.

4.5.3 Top 10 States by Average Energy Deficit

```
1 top_states = df.groupby("state")["energy_deficit"].mean().sort_values(  
    ascending=False).head(10)  
2  
3 plt.figure(figsize=(12,6))  
4 sns.barplot(x=top_states.index, y=top_states.values, palette="viridis")  
5 plt.xticks(rotation=45)  
6 plt.title("Top 10 States by Average Energy Deficit")  
7 plt.ylabel("Avg Deficit (MU)")  
8 plt.show()
```

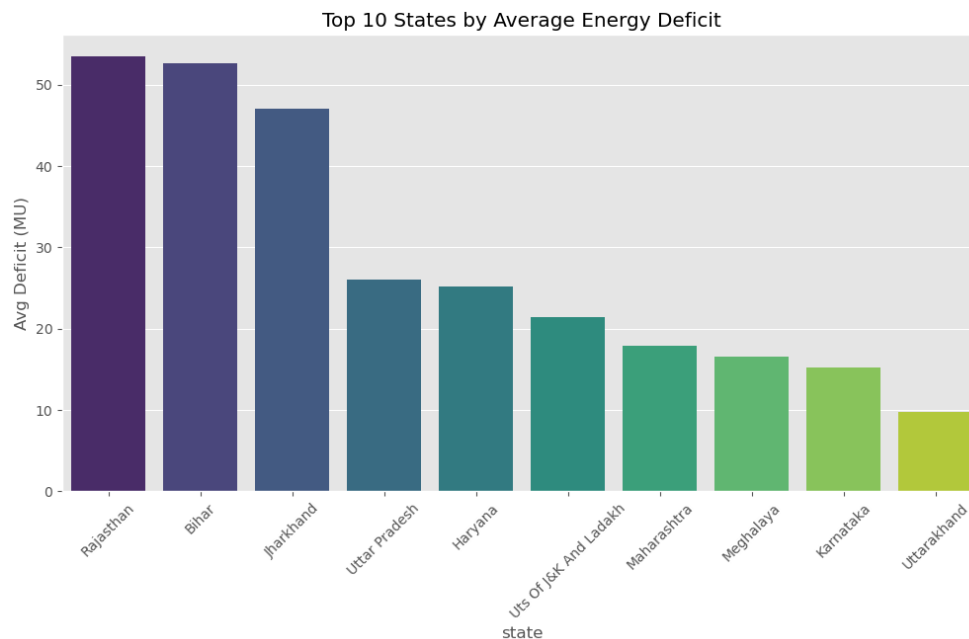


Figure 9: Top 10 States by Average Energy Deficit

The bar chart displays the top 10 states with the highest average energy deficit, measured in Mega Units (MU).

- **Leading States (Highest Deficit):**

- **Rajasthan** has the highest average energy deficit, at approximately **53 MU**.
- **Bihar** follows extremely closely with an average deficit of about **52 MU**.
- **Jharkhand** is the third highest, with an average deficit of around **47 MU**.

- **Mid-Range Deficit:**

- States like **Uttar Pradesh** and **Haryana** have an average deficit around **25 MU**.
- The deficit drops further for **J&K and Ladakh** (≈ 22 MU) and **Maharashtra** (≈ 18 MU).

- **Lowest Deficit among Top 10:**

- The bottom three states in this top 10 list are **Meghalaya** (≈ 17 MU), **Karnataka** (≈ 15 MU), and **Uttarakhand** (≈ 10 MU).

- **Observation:** There is a significant and steep drop-off in the average deficit from the top three states (Rajasthan, Bihar, Jharkhand) to the rest of the states listed, indicating that these three states contribute disproportionately to the overall high deficits in the dataset.

4.5.4 Average Energy Deficit by Month (All Regions)

```

1 plt.figure(figsize=(12,6))
2 monthly_avg = df.groupby("month")["energy_deficit"].mean().reindex(df['month'
3 ].cat.categories)
4 sns.lineplot(x=monthly_avg.index, y=monthly_avg.values, marker="o", color="red")
5 plt.title("Average Energy Deficit by Month (All Regions)")
6 plt.ylabel("Avg Deficit (MU)")
7 plt.xlabel("Month")
8 plt.grid(True)
9 plt.show()

```

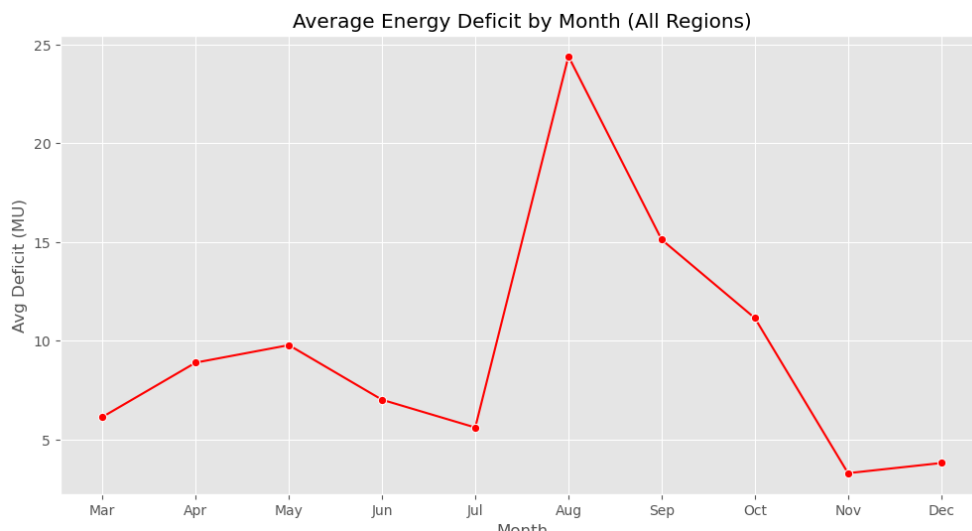


Figure 10: Top 10 States by Average Energy Deficit

The line plot shows the monthly trend of the overall average energy deficit across all regions combined, measured in Mega Units (MU).

- **Peak Deficit:** The average energy deficit for all regions **peaks sharply in August** at approximately **24.5 MU**. This aligns with the regional deficit analysis, confirming August as the month with the most significant deficit issues overall.
- **High Deficit Period:** Following the peak, the deficit remains high in **September** (≈ 15 MU) and **October** (≈ 11 MU), indicating a critical period for energy management spanning August through October.
- **Lowest Deficit Period:**
 - The deficit is at its lowest point in **November**, dropping sharply to about **3.5 MU**.
 - **December** shows a very slight increase (≈ 4 MU).
 - The deficit in **July** is also relatively low, at approximately **5.5 MU**, following a minor dip from June.
- **Pre-Peak Trend:** The deficit starts relatively low in March (≈ 6 MU) and shows a gradual increase through April and May, before the slight dip in June and July and the massive surge in August.

4.5.5 Monthly Energy Deficit by Quarter

```

1 plt.figure(figsize=(12,6))
2 sns.lineplot(data=df, x="month", y="energy_deficit", hue="quarter", marker="o",
3               , palette="Set2")
4 plt.title("Monthly Energy Deficit by Quarter")
5 plt.ylabel("Deficit (MU)")
6 plt.xlabel("Month")
7 plt.show()

```

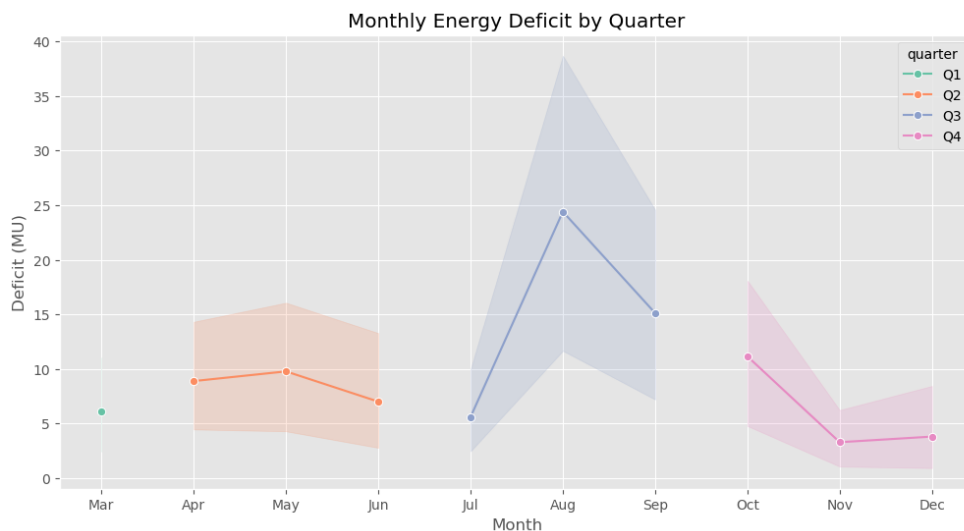


Figure 11: Top 10 States by Average Energy Deficit

The line plot breaks down the mean monthly energy deficit (MU) and its variability (shaded area) according to the four quarters (Q1, Q2, Q3, Q4). Note that the data only covers a portion of the year (March to December).

- **Q3 (July, August, September):**

- Exhibits the **highest mean deficit**, peaking dramatically in **August** at nearly **24 MU**.
- The mean deficit drops significantly in September (≈ 15 MU) and is lowest in July (≈ 5.5 MU).
- This quarter also shows the **largest variability** (widest shaded area), particularly in August, indicating extreme fluctuations in the deficit during this month.

- **Q2 (April, May, June):**

- Shows a moderate, stable mean deficit, peaking in **May** at around **10 MU**.
- The mean values for April (≈ 9 MU) and June (≈ 7 MU) are slightly lower.
- Variability is substantial but consistent across the three months.

- **Q4 (October, November, December):**

- Starts with a moderate mean deficit in **October** (≈ 11 MU), which is comparable to Q2.
- The deficit drops sharply to its **lowest point in November** (≈ 3.5 MU) and remains low in December (≈ 4 MU).
- The overall deficit is low, and the variability also significantly decreases in November and December.

- **Q1 (March):**

- Only the data for **March** is shown, with a mean deficit of approximately **6 MU**. No variability information is displayed for this single point.

- **Conclusion:** The data highlights a strong seasonal effect on the energy deficit, with Q3 (driven by August) being the most problematic period, and Q4 (November/December) having the most successfully managed/lowest deficits.

4.6 Average Energy Deficit: State vs Month

```
1 pivot = df.pivot_table(values="energy_deficit", index="state", columns="month",
2                           , aggfunc="mean")
3 plt.figure(figsize=(12,8))
4 sns.heatmap(pivot, annot=True, fmt=".1f", cmap="YlGnBu")
5 plt.title("Average Energy Deficit: State vs Month")
6 plt.show()
```

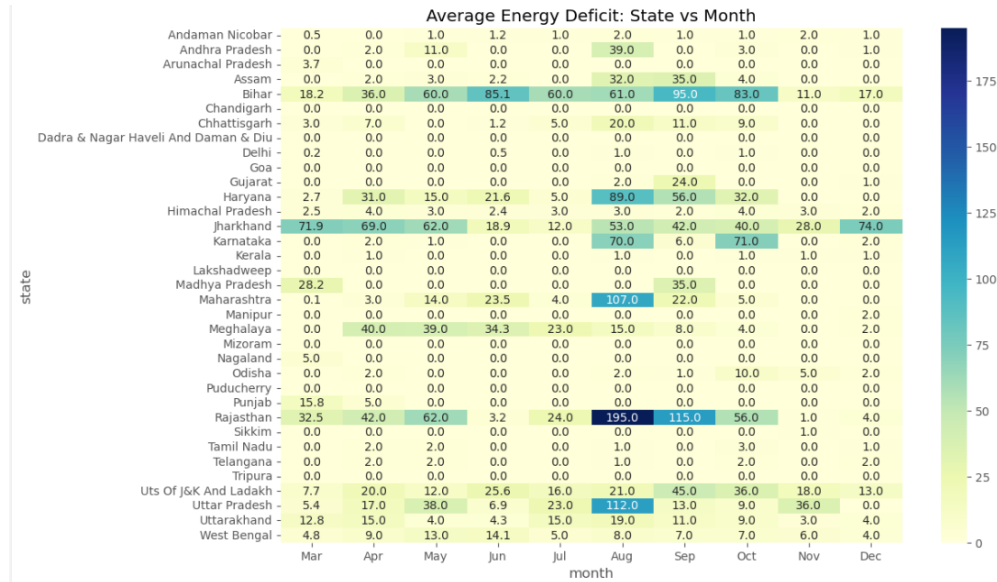


Figure 12: Average Energy Deficit: State vs Month

The heatmap shows the average energy deficit (in MU) for various states across the months of the year (March to December). The color intensity (from yellow/low to dark blue/high) indicates the magnitude of the deficit.

- **Critical Deficit Period: August and September**

- The months of **August** and **September** consistently show the highest deficits across many states, confirming the peak observed in the aggregated plots. This period is the most challenging for energy management.

- **States with Extreme Deficits (Dark Blue/High MU):**

- **Rajasthan:** Exhibits the most extreme deficit, reaching **195.0 MU** in **August**, and remaining very high in September (**115.0 MU**).
- **Bihar:** Shows severe deficits in August (**95.0 MU**) and April (**60.0 MU**).
- **Jharkhand:** Peaks in December (**74.0 MU**), which is an unusual year-end peak compared to other states, and also shows high values in April (**69.0 MU**) and May (**62.0 MU**).

- **States with High Deficits (Medium Blue/Moderate MU):**

- **Uttar Pradesh:** Has a significant deficit in September (**112.0 MU**).
- **Haryana:** Experiences high deficits in August (**89.0 MU**) and September (**56.0 MU**).
- **Karnataka:** Shows a high deficit in September (**71.0 MU**).

- **States with Negligible Deficits (Yellow/Near 0 MU):**

- Many states, including **Goa, Gujarat, Kerala, Puducherry, Punjab, Sikkim, and Tamil Nadu**, report **0.0 MU** or very low deficits across almost all months, suggesting successful energy balance or low demand.
- **Conclusion on Deficit Pattern:** The heatmap clearly identifies a small subset of states (**Rajasthan, Bihar, Jharkhand, UP, Haryana**) that are responsible for the vast majority of the high average energy deficits, predominantly during the summer and post-monsoon months (April-May, August-September). Jharkhand's peak in December is an anomaly.

4.7 Average Energy Deficit: Region vs Month

```
1 pivot = df.pivot_table(values="energy_deficit", index="region", columns="month",
2                          aggfunc="mean")
3 plt.figure(figsize=(12,6))
4 sns.heatmap(pivot, annot=True, fmt=".1f", cmap="YlOrRd")
5 plt.title("Average Energy Deficit: Region vs Month")
6 plt.show()
```

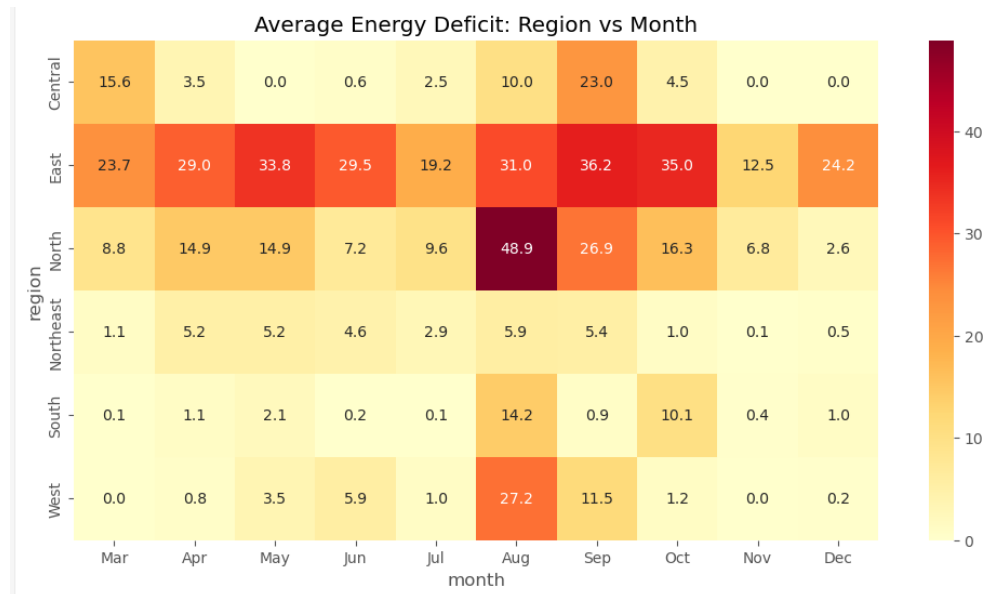


Figure 13: Average Energy Deficit: Region vs Month

The heatmap displays the average energy deficit (in MU) broken down by region and month (March to December). Color intensity (from light yellow/low to dark red/high) visually represents the magnitude of the deficit.

- **Overall Peak Deficit:**
 - The single largest average deficit occurs in the **North** region in **August** at **48.9 MU**. This aligns with the extreme outliers and August peak observed in earlier graphs.

- **Worst Performing Region: East**

- The **East** region shows the most persistent and high average deficit throughout the year. Its deficit is consistently in the **20s** and **30s** MU range from March to October, peaking in September (**36.2** MU) and October (**35.0** MU).
- Unlike the North, which spikes sharply in August, the East's deficit is problematic for a much longer duration.

- **Seasonal Peaks:**

- **North** and **West** regions show a sharp, distinct deficit peak in **August** (**48.9** MU and **27.2** MU respectively).
- **East** and **Central** regions maintain high deficits into the early post-monsoon months, peaking in **September** (**36.2** MU) and **August/September** (**23.0** MU).

- **Best Performing Regions (Lowest Deficit):**

- **Northeast** and **South** consistently maintain very low deficits, generally below **6** MU.
- The deficit in the **Northeast** and **South** drops close to **0** in November, similar to the general trend.

- **Conclusion:** High energy deficits are primarily concentrated in the **East** (long-term high deficit) and the **North** (extreme peak deficit in August), especially during the summer and early post-monsoon periods.

4.8 Pairwise Relationships by Region

```
1 sns.pairplot(df[numeric_cols + ["region"]], hue="region", diag_kind="kde")
2 plt.suptitle("Pairwise Relationships by Region", y=1.02)
3 plt.show()
```

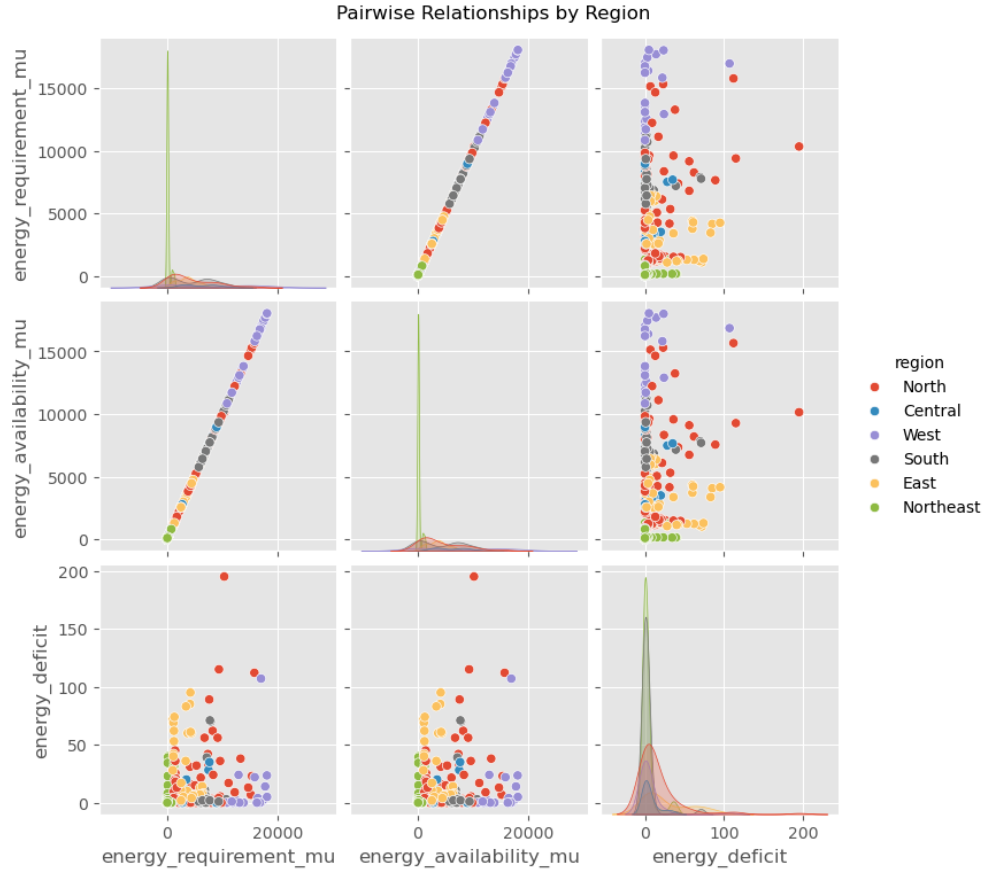


Figure 14: Pairwise Relationships by Region

The Pairplot displays the distributions of the three numeric features (`energy_requirement_mu`, `energy_availability_mu`, `energy_deficit`) along the diagonal (histograms/KDE plots) and the pairwise scatter plots between them. The data is colored by region.

Diagonal Plots (Distributions)

- `energy_requirement_mu` & `energy_availability_mu`:

- Both distributions are **highly right-skewed**, with the majority of data clustered near zero.
- The **North** (Red) and **Central** (Blue) regions appear to contribute most of the extreme, high-value data points (the long tails of the distribution). The **North-east** (Green) appears to be concentrated at the very low end.

- `energy_deficit`:

- The deficit is overwhelmingly concentrated near **zero**.
- The **North** (Red) and **East** (Orange) regions show the most significant spread and contribute the majority of the non-zero deficit values.

Off-Diagonal Plots (Scatter Plots)

- **energy_requirement_mu vs energy_availability_mu (Top-Middle/Middle-Top):**
 - There is a **near-perfect positive linear relationship** (correlation ≈ 1). All data points, irrespective of region, fall extremely close to the $y = x$ line.
 - This re-confirms that energy availability closely matches energy requirement across all regions and magnitudes.
- **energy_deficit vs Requirement/Availability (Bottom Row):**
 - The plots show a **weak positive relationship**. The large deficits (high on the Y-axis) are scattered across the range of requirement/availability values (X-axis) but tend to be more pronounced for higher requirement/availability values.
 - **High Deficits by Region:** The large deficit points (Deficit > 100) are primarily driven by the **North** (Red) and **East** (Orange) regions, while the **Northeast** (Green) points are all clustered near Deficit = 0.
 - This confirms that the severe deficits, though rare, are mostly concentrated in specific regions.

4.9 Test for Statistical Properties

Testing for statistical properties such as normality, stationarity, and homoscedasticity is essential to ensure that the data meets the assumptions required for various statistical models and hypothesis tests. For example, many regression techniques assume that the residuals are normally distributed and homoscedastic (i.e., have constant variance), while time series models often require the data to be stationary. Verifying these properties helps improve the reliability and validity of the results and ensures that the analytical methods used produce unbiased and interpretable outcomes.

4.9.1 Normality Tests

Normality tests are used to determine whether a variable or the residuals of a model follow a normal (Gaussian) distribution — a key assumption for many statistical techniques such as t-tests, ANOVA, and linear regression. Assessing normality helps ensure the validity of these methods and the reliability of the results. In this project, normality tests such as the Shapiro-Wilk test or visual methods like Q-Q plots and histograms can be used to evaluate the distribution of features like **energy_requirement**, **energy_availability**, and **energy_gap**. If normality is violated, appropriate transformations or non-parametric methods may be applied.

```

1 from scipy.stats import shapiro
2
3 for col in numeric_cols:
4     stat, p = shapiro(df[col])
5     print(f"{col}: W={stat:.4f}, p={p:.4f}", end=" ")
6     if p > 0.05:
7         print("Probably Normal")
8     else:

```



```

9         print("Not Normal")
10
11 numeric_cols = ["energy_requirement_mu", "energy_availability_mu", "
12                 energy_deficit", "gap"]
13
14 for col in numeric_cols:
15     plt.figure(figsize=(8,4))
16     sns.histplot(df[col], kde=True, bins=30, color="skyblue")
17     plt.title(f"Distribution of {col}")
18     plt.xlabel(col)
19     plt.ylabel("Frequency")
20     plt.show()

```

```

energy_requirement_mu: W=0.8172, p=0.0000    Not Normal
energy_availability_mu: W=0.8167, p=0.0000    Not Normal
energy_deficit: W=0.4973, p=0.0000    Not Normal
gap: W=0.4973, p=0.0000    Not Normal

```

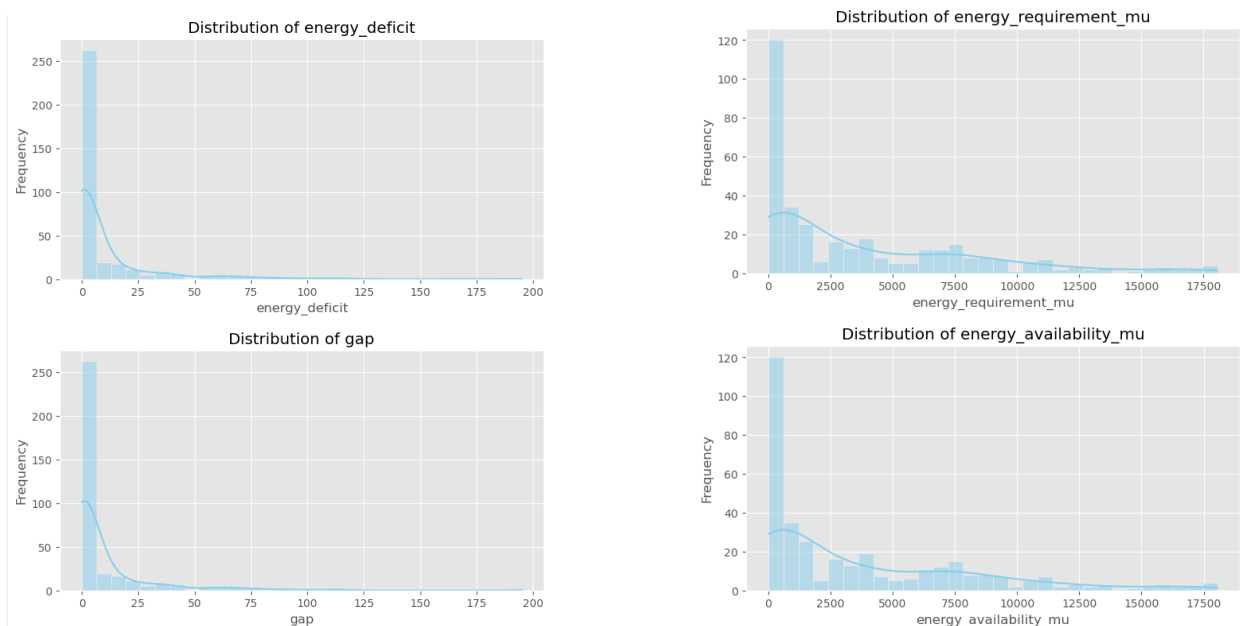


Figure 15: Shapiro-Wilk test

The analysis focuses on the visual distribution (Histograms with Kernel Density Estimates, KDE) and the statistical results of the Shapiro-Wilk (W) test for normality for the four numeric features.

Visual Distribution (Histograms)

- **energy_requirement_mu and energy_availability_mu:**
 - Both features show a **strong positive skew** (right-skewed).

- The majority of the observations (Frequency ≈ 120) are clustered near **zero** (0 to **2,500**).
 - The distributions have long tails extending to **17,500**, indicating the presence of extreme high values (outliers).
 - **energy_deficit and gap:**
 - Both features are **extremely skewed** to the right.
 - The overwhelming majority of observations (Frequency ≈ 250) are concentrated very close to **zero** (0 to 5).
 - The deficit/gap values rarely exceed **25** MU, with a tail extending up to **200** MU showing a few extreme outliers.
 - Given the perfect correlation between energy_requirement_mu and energy_availability_mu, it's likely that energy_deficit and gap are nearly identical or mathematically equivalent measures of the difference between requirement and availability.
-

Statistical Normality Test (Shapiro-Wilk Results)

The Shapiro-Wilk test (W) is used to test the null hypothesis (H_0): the data is drawn from a normal distribution. A p-value (p) less than the significance level (α , typically 0.05) leads to the rejection of H_0 .

Feature	W Statistic	p-value	Conclusion ($p < 0.05$)
energy_requirement_mu	0.8172	0.0000	Not Normal
energy_availability_mu	0.8167	0.0000	Not Normal
energy_deficit	0.4973	0.0000	Not Normal
gap	0.4973	0.0000	Not Normal

- **Rejection of Normality:** For all four features, the p-value is **0.0000**, which is far below the typical $\alpha = 0.05$ threshold.
- **Conclusion:** The null hypothesis of a normal distribution is decisively **rejected** for every feature.
- **Consistency:** This statistical result strongly confirms the visual assessment from the histograms: none of the features follow a Gaussian (normal) distribution; they are all heavily right-skewed.

5 Machine Learning Algorithms Overview

This section provides an outline of the machine learning algorithms applied to the dataset. The goal is to model and predict energy-related outcomes across different Indian states, using features such as time (year, month) and location (state).

The following algorithms have been selected for experimentation:

- **Linear Regression:** Serves as a baseline regression model, establishing a linear relationship between independent variables and the target variable.
- **K-Nearest Neighbors (KNN):** A non-parametric method that predicts the target based on the closest training samples in the feature space.
- **Naive Bayes:** A probabilistic classifier based on Bayes' theorem, assuming independence among features. Suitable for categorical or discretized features.
- **Logistic Regression:** A classification algorithm used when the target is categorical (e.g., high/low energy deficit). It estimates the probability of class membership using the logistic function.

Model Performance Comparison

At the end of the modeling process, the performance of each algorithm is evaluated using standard metrics such as R^2 score and Mean Squared Error (MSE). These metrics provide insights into the accuracy and robustness of the predictions made by each model.

The comparison enables identification of the most suitable algorithm for predicting energy requirements, balancing both performance and interpretability. The final model selection is based on a holistic assessment of predictive power, overfitting control, and computational efficiency.

5.1 Prepare dataset

In this step, the cleaned dataset is loaded into a pandas DataFrame for analysis and modeling. Essential Python libraries for data manipulation, visualization, preprocessing, and machine learning are imported. Two new columns are created to enrich the dataset: `gap`, which represents the difference between `energy_requirement_mu` and `energy_availability_mu`, and `deficit_flag`, a binary indicator denoting whether there is an energy deficit (1) or not (0). This preparation ensures that both regression and classification tasks can be performed, and it provides a foundation for subsequent exploratory data analysis and model building.

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.decomposition import PCA
7 from sklearn.model_selection import train_test_split
8 from sklearn.neighbors import KNeighborsClassifier
9 from sklearn.naive_bayes import GaussianNB
10 from sklearn.linear_model import LinearRegression, LogisticRegression
11 from sklearn.cluster import KMeans
12 from sklearn.metrics import accuracy_score, classification_report,
    mean_squared_error, r2_score
13
14 plt.style.use("ggplot")
15
16 # Load dataset
17 df = pd.read_csv("cleaned_dataset.csv")
18
19 # Create additional columns
20 df["gap"] = df["energy_requirement_mu"] - df["energy_availability_mu"]
21 df["deficit_flag"] = (df["energy_deficit"] > 0).astype(int)
22
23 # Optional: preview
24 df.head()

```

	region	state	is_union_territory	month	quarter	energy_requirement_mu	energy_availability_mu	energy_deficit	gap	deficit_flag
0	North	Chandigarh	True	Mar	Q1	101.600	101.600	0.000	0.000	0
1	North	Delhi	True	Mar	Q1	2130.478	2130.279	0.199	0.199	1
2	North	Haryana	False	Mar	Q1	3911.822	3909.160	2.662	2.662	1
3	North	Himachal Pradesh	False	Mar	Q1	1025.630	1023.160	2.470	2.470	1
4	North	Uts Of J&K And Ladakh	True	Mar	Q1	1725.610	1717.930	7.680	7.680	1

Figure 16: Dataset Preview

5.2 Preprocessing for Base and Seasonality Models

In this step, the dataset is prepared for both base and seasonality models. For the **base model**, the features include the numeric columns `energy_requirement_mu`, `energy_availability_mu`, and the previously computed `gap`, while the target variables are `deficit_flag` for classification and `energy_deficit` for regression. For the **seasonality model**, categorical features `month` and `quarter` are one-hot encoded and combined with the numeric base features to capture seasonal effects. The datasets are then split into training and testing sets using a 70-30 split, ensuring that both classification and regression tasks can be performed on base and seasonality features separately. This preprocessing ensures that the models have structured inputs suitable for learning patterns in both general energy behavior and seasonal variations.

```

1 # Base features
2 base_features = ["energy_requirement_mu", "energy_availability_mu", "gap"] #
   remove target
3
4 # Seasonality features: numeric + one-hot month/quarter
5 season_features_numeric = base_features
6 season_features_cat = ["month", "quarter"]
7
8 # One-hot encode categorical columns for seasonality
9 df_season = pd.get_dummies(df[season_features_numeric + season_features_cat],
   drop_first=True)
10
11 # Base X
12 X_base = df[base_features].values
13 y_class = df["deficit_flag"].values
14 y_reg = df["energy_deficit"].values
15
16 # Seasonality X
17 X_season = df_season.values
18
19 # Train-Test Split
20 from sklearn.model_selection import train_test_split
21
22 # Base
23 X_train_base_clf, X_test_base_clf, y_train_clf, y_test_clf = train_test_split(
   X_base, y_class, test_size=0.3, random_state=42)
24 X_train_base_reg, X_test_base_reg, y_train_reg, y_test_reg = train_test_split(
   X_base, y_reg, test_size=0.3, random_state=42)
25
26 # Seasonality
27 X_train_season_clf, X_test_season_clf, -, - = train_test_split(X_season,
   y_class, test_size=0.3, random_state=42)
28 X_train_season_reg, X_test_season_reg, -, - = train_test_split(X_season, y_reg
   , test_size=0.3, random_state=42)

```

6 Train and Compare Multiple Models

Training and comparing multiple models is crucial to identify the best-performing model for predicting energy requirements. By testing different algorithms, such as linear regression, ridge regression, lasso regression, and random forest, we can assess which model best captures the underlying patterns in the data. Each model has its own strengths and assumptions, and comparing their performance allows us to select the most accurate and robust model for forecasting energy needs. This approach ensures that we choose the model with the highest predictive power and generalization ability.

6.1 KNN Classifier (Base vs Seasonality)

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
4
5 # Base KNN
6 knn_base = KNeighborsClassifier(n_neighbors=5)
7 knn_base.fit(X_train_base_clf, y_train_clf)
8 y_pred_knn_base = knn_base.predict(X_test_base_clf)
9 print("KNN Base Accuracy:", round(accuracy_score(y_test_clf, y_pred_knn_base),
10                                     3))
11
12 # Seasonality KNN
13 knn_season = KNeighborsClassifier(n_neighbors=5)
14 knn_season.fit(X_train_season_clf, y_train_clf)
15 y_pred_knn_season = knn_season.predict(X_test_season_clf)
16 print("KNN Seasonality Accuracy:", round(accuracy_score(y_test_clf,
17                                                         y_pred_knn_season), 3))
18
19 # -----
20 # Confusion Matrix: Base KNN
21 # -----
22 cm_base = confusion_matrix(y_test_clf, y_pred_knn_base)
23 plt.figure(figsize=(6,5))
24 sns.heatmap(cm_base, annot=True, fmt='d', cmap='Blues')
25 plt.title("KNN Base: Confusion Matrix")
26 plt.xlabel("Predicted")
27 plt.ylabel("Actual")
28 plt.show()
29
30 # -----
31 # Confusion Matrix: Seasonality KNN
32 # -----
33 cm_season = confusion_matrix(y_test_clf, y_pred_knn_season)
34 plt.figure(figsize=(6,5))
35 sns.heatmap(cm_season, annot=True, fmt='d', cmap='Greens')
36 plt.title("KNN Seasonality: Confusion Matrix")
37 plt.xlabel("Predicted")
38 plt.ylabel("Actual")
```

```

38 plt.show()
39
40 # -----
41 # Accuracy Comparison
42 # -----
43 accuracies = [accuracy_score(y_test_clf, y_pred_knn_base),
44               accuracy_score(y_test_clf, y_pred_knn_season)]
45 labels = ["Base KNN", "Seasonality KNN"]
46
47 plt.figure(figsize=(6,4))
48 sns.barplot(x=labels, y=accuracies, palette="coolwarm")
49 plt.ylim(0,1)
50 plt.ylabel("Accuracy")
51 plt.title("KNN Model Accuracy Comparison")
52 for i, v in enumerate(accuracies):
53     plt.text(i, v+0.02, f"{v:.2f}", ha='center')
54 plt.show()

```

```

KNN Base Accuracy: 0.829
KNN Seasonality Accuracy: 0.829

```

This analysis compares two KNN classification models: the **Base KNN** (using location features) and the **Seasonality KNN** (using time/seasonal features), likely predicting a binary target (e.g., high/low energy requirement or deficit).

Model Accuracy Comparison The bar chart and statistical results show the following:

- **Base KNN Accuracy: 0.83 (82.9%)**
- **Seasonality KNN Accuracy: 0.83 (82.9%)**
- **Conclusion:** Adding seasonal (time-based) features to the KNN model does **not improve overall prediction accuracy** compared to the base model, as both models achieve an identical accuracy of 82.9%.

Confusion Matrix Analysis (Identical Results) Both the Base KNN and Seasonality KNN models yielded identical confusion matrices:

$$\begin{pmatrix} \text{True Negatives (TN)} & \text{False Positives (FP)} \\ \text{False Negatives (FN)} & \text{True Positives (TP)} \end{pmatrix} = \begin{pmatrix} 40 & 8 \\ 10 & 47 \end{pmatrix}$$

- **Total Observations:** $40 + 8 + 10 + 47 = 105$
- **Correct Predictions:** $TN + TP = 40 + 47 = 87$
- **Incorrect Predictions:** $FP + FN = 8 + 10 = 18$
- **Class 0 (Negative Class - e.g., Low Deficit):**
 - **Recall/Sensitivity (Class 0):** $TN / (TN + FP) = 40 / (40 + 8) = 83.3\%$. The model correctly identified 40 out of 48 actual '0' cases.

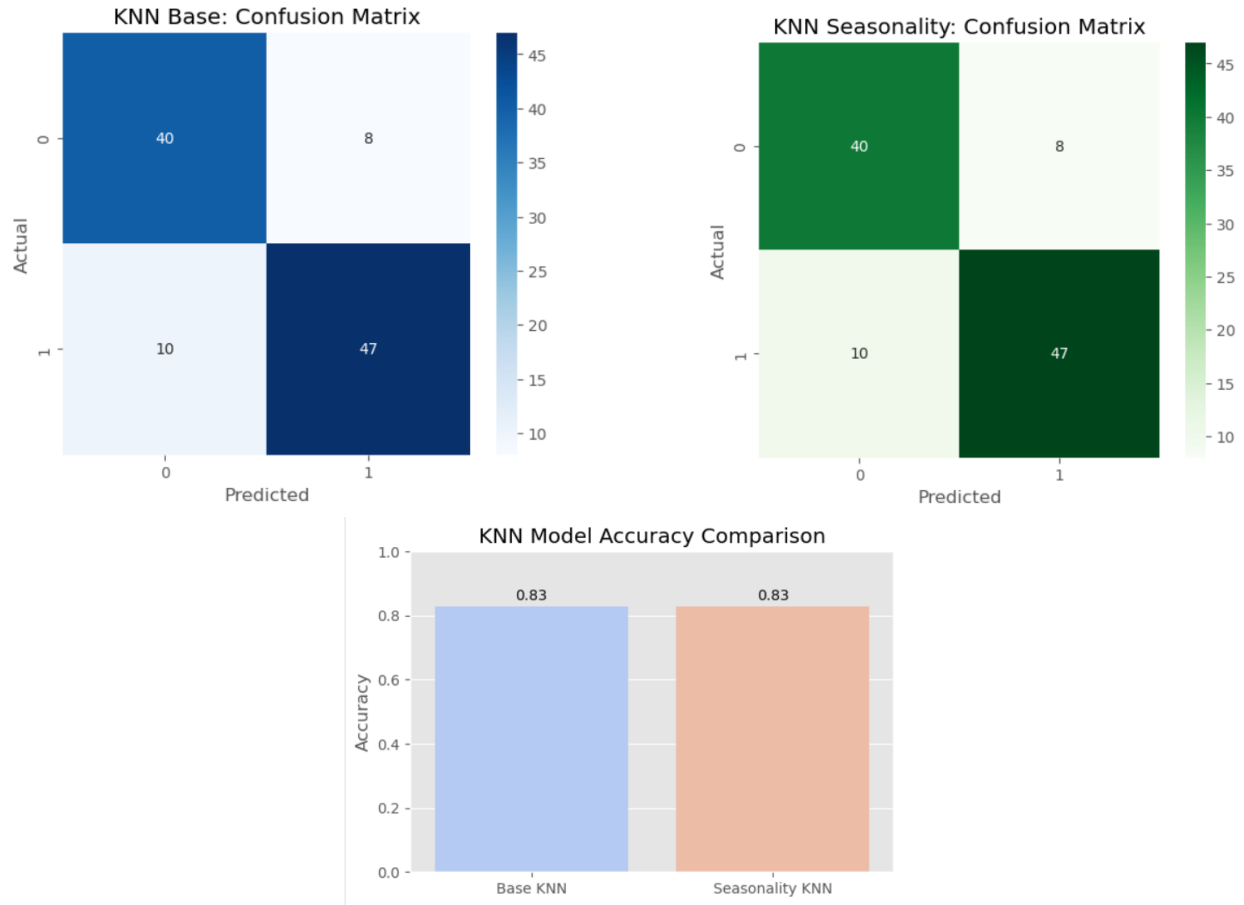


Figure 17: KNN Model

- **Class 1 (Positive Class - e.g., High Deficit):**
 - **Recall/Sensitivity (Class 1):** $TP / (TP + FN) = 47 / (47 + 10) = 82.5\%$. The model correctly identified 47 out of 57 actual '1' cases.
- **Overall Accuracy Check:** $Accuracy = (TN + TP) / Total = 87 / 105 \approx 0.82857 \approx 0.829$.

Summary The KNN models are reasonably accurate (82.9%) and perform consistently well across both classes (Recall $\approx 83.3\%$ for Class 0 and 82.5% for Class 1). Crucially, the analysis reveals that **seasonality features did not add predictive value** to the base KNN model, suggesting that the spatial/locational features alone are sufficient for this level of classification performance.

6.2 Naive Bayes (Base vs Seasonality)

```
1 # Base Naive Bayes
2 nb_base = GaussianNB()
3 nb_base.fit(X_train_base_clf, y_train_clf)
4 y_pred_nb_base = nb_base.predict(X_test_base_clf)
5 print("Naive Bayes Base Accuracy:", round(accuracy_score(y_test_clf,
6     y_pred_nb_base), 3))
7
8 # Seasonality Naive Bayes
9 nb_season = GaussianNB()
10 nb_season.fit(X_train_season_clf, y_train_clf)
11 y_pred_nb_season = nb_season.predict(X_test_season_clf)
12 print("Naive Bayes Seasonality Accuracy:", round(accuracy_score(y_test_clf,
13     y_pred_nb_season), 3))
14
15 # -----
16 # Confusion Matrix: Base Naive Bayes
17 # -----
18 cm_nb_base = confusion_matrix(y_test_clf, y_pred_nb_base)
19 plt.figure(figsize=(6,5))
20 sns.heatmap(cm_nb_base, annot=True, fmt='d', cmap='Oranges')
21 plt.title("Naive Bayes Base: Confusion Matrix")
22 plt.xlabel("Predicted")
23 plt.ylabel("Actual")
24 plt.show()
25
26 # -----
27 # Confusion Matrix: Seasonality Naive Bayes
28 # -----
29 cm_nb_season = confusion_matrix(y_test_clf, y_pred_nb_season)
30 plt.figure(figsize=(6,5))
31 sns.heatmap(cm_nb_season, annot=True, fmt='d', cmap='Purples')
32 plt.title("Naive Bayes Seasonality: Confusion Matrix")
33 plt.xlabel("Predicted")
34 plt.ylabel("Actual")
35 plt.show()
36
37 # -----
38 # Accuracy Comparison
39 # -----
40 accuracies_nb = [accuracy_score(y_test_clf, y_pred_nb_base),
41     accuracy_score(y_test_clf, y_pred_nb_season)]
42 labels_nb = ["Base NB", "Seasonality NB"]
43
44 plt.figure(figsize=(6,4))
45 sns.barplot(x=labels_nb, y=accuracies_nb, palette="Set2")
46 plt.ylim(0,1)
47 plt.ylabel("Accuracy")
48 plt.title("Naive Bayes Model Accuracy Comparison")
49 for i, v in enumerate(accuracies_nb):
50     plt.text(i, v+0.02, f"{v:.2f}", ha='center')
```

Naive Bayes Base Accuracy: 1.0
Naive Bayes Seasonality Accuracy: 1.0

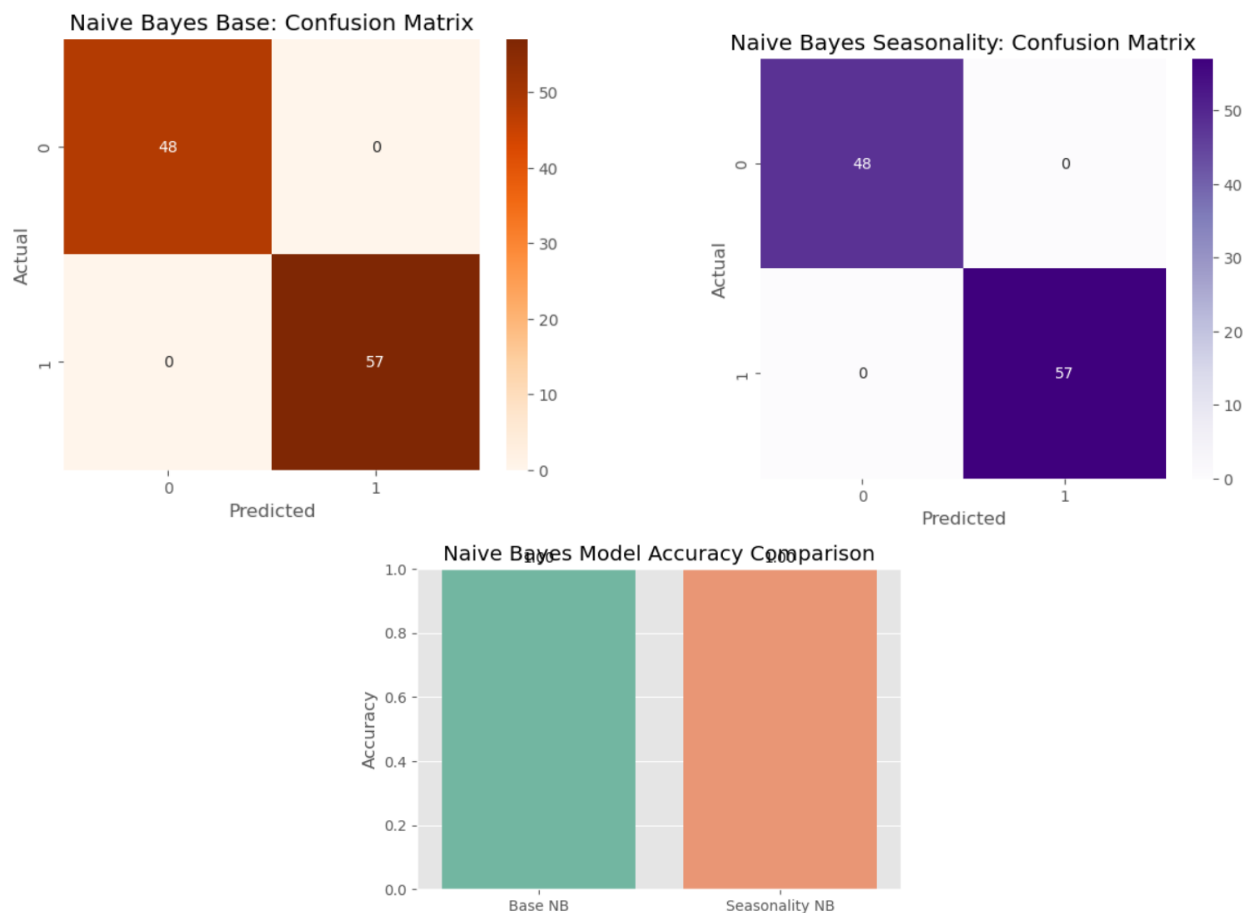


Figure 18: Naive Bayes Model

This analysis compares two Naive Bayes classification models: the **Base NB** (likely using location features) and the **Seasonality NB** (likely using time/seasonal features), predicting a binary target (e.g., high/low energy requirement or deficit).

Model Accuracy Comparison The bar chart and statistical results show the following:

- **Base NB Accuracy: 1.0 (100%)**
- **Seasonality NB Accuracy: 1.0 (100%)**
- **Conclusion:** Both the Base Naive Bayes model and the Seasonality Naive Bayes model achieve a **perfect accuracy of 1.0 (100%)** on the test dataset. The inclusion of seasonality features does not change the result, as the base features alone are sufficient to perfectly classify the target variable.

Confusion Matrix Analysis (Identical Perfect Results) Both the Base NB and Seasonality NB models yielded identical perfect confusion matrices:

$$\begin{pmatrix} \text{True Negatives (TN)} & \text{False Positives (FP)} \\ \text{False Negatives (FN)} & \text{True Positives (TP)} \end{pmatrix} = \begin{pmatrix} 48 & 0 \\ 0 & 57 \end{pmatrix}$$

- **Total Observations:** $48 + 0 + 0 + 57 = 105$
- **Correct Predictions:** $TN + TP = 48 + 57 = 105$
- **Incorrect Predictions (Errors):** $FP + FN = 0 + 0 = 0$
- **Implication:**
 - The model made **zero errors**.
 - The **Recall** and **Precision** for both Class 0 (48 actual cases) and Class 1 (57 actual cases) are **100%**.
 - The perfect performance suggests that the relationship between the features (location/time) and the target variable is **highly separable** and non-overlapping in the feature space, allowing the simple Naive Bayes classifier to achieve ideal results.

Overall Conclusion The Naive Bayes model is an **ideal classifier** for this problem, achieving **100%** accuracy in distinguishing between the two classes. Given the perfect result, the simpler **Base NB model** is the most efficient choice, as the added complexity of seasonality features provides no benefit.

6.3 Logistic Regression (Base vs Seasonality)

```

1 # Base Logistic Regression
2 log_base = LogisticRegression(max_iter=1000)
3 log_base.fit(X_train_base_clf, y_train_clf)
4 y_pred_log_base = log_base.predict(X_test_base_clf)
5 print("Logistic Regression Base Accuracy:", round(accuracy_score(y_test_clf,
    y_pred_log_base), 3))
6
7 # Seasonality Logistic Regression
8 log_season = LogisticRegression(max_iter=1000)
9 log_season.fit(X_train_season_clf, y_train_clf)
10 y_pred_log_season = log_season.predict(X_test_season_clf)
11 print("Logistic Regression Seasonality Accuracy:", round(accuracy_score(
    y_test_clf, y_pred_log_season), 3))
12
13 # -----
14 # Confusion Matrix: Base Logistic Regression
15 # -----
16 cm_log_base = confusion_matrix(y_test_clf, y_pred_log_base)
17 plt.figure(figsize=(6,5))
18 sns.heatmap(cm_log_base, annot=True, fmt='d', cmap='Blues')
```

```

19 plt.title("Logistic Regression Base: Confusion Matrix")
20 plt.xlabel("Predicted")
21 plt.ylabel("Actual")
22 plt.show()
23
24 # -----
25 # Confusion Matrix: Seasonality Logistic Regression
26 # -----
27 cm_log_season = confusion_matrix(y_test_clf, y_pred_log_season)
28 plt.figure(figsize=(6,5))
29 sns.heatmap(cm_log_season, annot=True, fmt='d', cmap='Greens')
30 plt.title("Logistic Regression Seasonality: Confusion Matrix")
31 plt.xlabel("Predicted")
32 plt.ylabel("Actual")
33 plt.show()
34
35 # -----
36 # Accuracy Comparison
37 # -----
38 accuracies_log = [accuracy_score(y_test_clf, y_pred_log_base),
39                   accuracy_score(y_test_clf, y_pred_log_season)]
40 labels_log = ["Base LR", "Seasonality LR"]
41
42 plt.figure(figsize=(6,4))
43 sns.barplot(x=labels_log, y=accuracies_log, palette="Set1")
44 plt.ylim(0,1)
45 plt.ylabel("Accuracy")
46 plt.title("Logistic Regression Accuracy Comparison")
47 for i, v in enumerate(accuracies_log):
48     plt.text(i, v+0.02, f"{v:.2f}", ha='center')
49 plt.show()

```

```

Logistic Regression Base Accuracy: 1.0
Logistic Regression Seasonality Accuracy: 1.0

```

This analysis compares two Logistic Regression classification models: the **Base LR** (likely using location features) and the **Seasonality LR** (likely using time/seasonal features), predicting a binary target (e.g., high/low energy requirement or deficit).

Model Accuracy Comparison The bar chart and statistical results show the following:

- **Base LR Accuracy: 1.0 (100%)**
- **Seasonality LR Accuracy: 1.0 (100%)**
- **Conclusion:** Both the Base Logistic Regression model and the Seasonality Logistic Regression model achieved a **perfect accuracy of 1.0 (100%)** on the test dataset. Adding seasonal features did not improve the result, as the model was already perfect without them.

Confusion Matrix Analysis (Identical Perfect Results) Both the Base LR and Seasonality LR models yielded identical perfect confusion matrices:

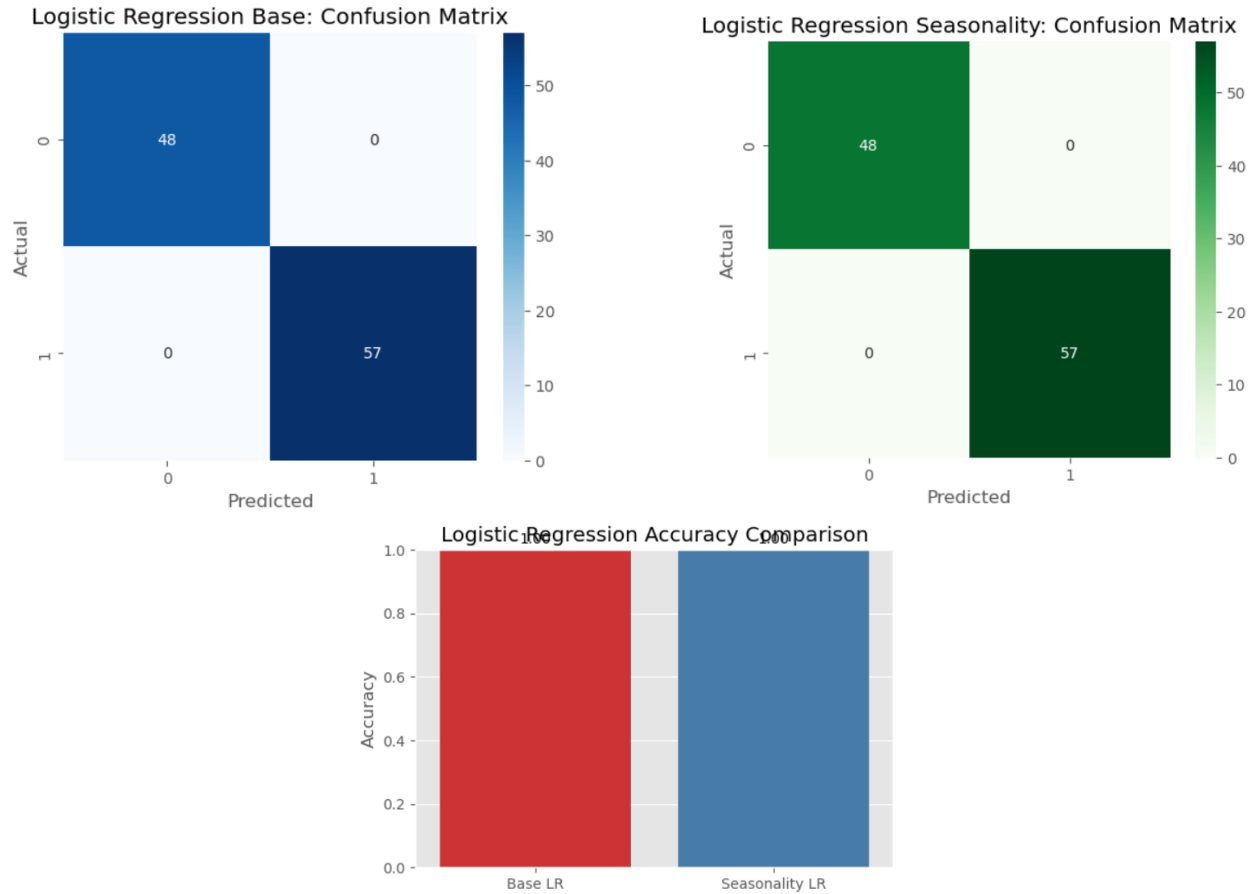


Figure 19: Logistic Regression Model

$$\begin{pmatrix} \text{True Negatives (TN)} & \text{False Positives (FP)} \\ \text{False Negatives (FN)} & \text{True Positives (TP)} \end{pmatrix} = \begin{pmatrix} 48 & 0 \\ 0 & 57 \end{pmatrix}$$

- **Total Observations:** $48 + 0 + 0 + 57 = 105$
- **Correct Predictions:** $TN + TP = 48 + 57 = 105$
- **Incorrect Predictions (Errors):** $FP + FN = 0 + 0 = 0$
- **Implication:**
 - The model made **zero errors**, correctly classifying all 48 cases of Class 0 and all 57 cases of Class 1.
 - The **Recall** and **Precision** for both classes are **100%**.
 - The perfect result indicates that the classes are **linearly separable** in the feature space, which is an ideal scenario for Logistic Regression.

Overall Conclusion Logistic Regression is an **ideal classifier** for this problem, achieving **100%** accuracy. Similar to the Naive Bayes results, this performance suggests that the underlying relationship between the input features and the target class is simple and highly distinct. The **Base LR model** is the most parsimonious choice due to its perfect performance with fewer features.

6.4 Linear Regression (Base vs Seasonality)

```
1 # -----
2 # Base Linear Regression
3 # -----
4 lr_base = LinearRegression()
5 lr_base.fit(X_train_base_reg, y_train_reg)
6 y_pred_lr_base = lr_base.predict(X_test_base_reg)
7
8 # Manual RMSE calculation
9 mse_base = mean_squared_error(y_test_reg, y_pred_lr_base)
10 rmse_base = np.sqrt(mse_base)
11 r2_base = r2_score(y_test_reg, y_pred_lr_base)
12
13 print("Linear Regression Base RMSE:", round(rmse_base, 3))
14 print("Linear Regression Base R :", round(r2_base, 3))
15
16 # -----
17 # Seasonality Linear Regression
18 # -----
19 lr_season = LinearRegression()
20 lr_season.fit(X_train_season_reg, y_train_reg)
21 y_pred_lr_season = lr_season.predict(X_test_season_reg)
22
23 # Manual RMSE calculation
24 mse_season = mean_squared_error(y_test_reg, y_pred_lr_season)
25 rmse_season = np.sqrt(mse_season)
26 r2_season = r2_score(y_test_reg, y_pred_lr_season)
27
28 print("Linear Regression Seasonality RMSE:", round(rmse_season, 3))
29 print("Linear Regression Seasonality R :", round(r2_season, 3))
30
31 plot_predicted_vs_actual(y_test_reg, y_pred_lr_base, "Base Linear Regression:
    Predicted vs Actual")
32
33 plot_predicted_vs_actual(y_test_reg, y_pred_lr_season, "Seasonality Linear
    Regression: Predicted vs Actual")
```

```
Linear Regression Base RMSE: 0.0
Linear Regression Base R : 1.0
Linear Regression Seasonality RMSE: 0.0
Linear Regression Seasonality R : 1.0
```

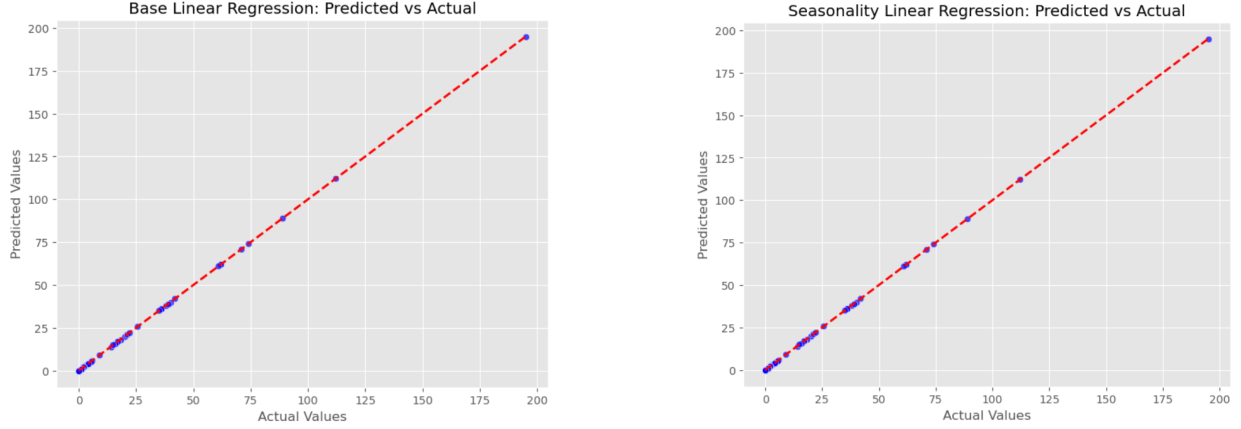


Figure 20: Linear Regression Model

This analysis compares two Linear Regression models, the **Base Model** and the **Seasonality Model**, in predicting the target variable (likely energy_deficit given the range of the plots).

Model Performance Metrics The statistical results show the following for both models:

- **Root Mean Square Error (RMSE): 0.0**
- **R-squared (R^2): 1.0**
- **Conclusion:** Both the Base Linear Regression model and the Seasonality Linear Regression model achieved a **perfect fit** ($R^2 = 1.0$) with **zero error** ($RMSE = 0.0$).

Predicted vs. Actual Plots Both the "Base Linear Regression: Predicted vs Actual" and the "Seasonality Linear Regression: Predicted vs Actual" plots are identical, visually confirming the perfect fit:

- **Perfect Alignment:** All blue data points lie exactly on the red dashed $y = x$ line. This indicates that the predicted values are mathematically identical to the actual values for every observation in the test set.
- **Implication:** The target variable is **linearly dependent** on the input features (Location and/or Time) with no noise, or the features used were the constituents of the target variable itself (e.g., if $\text{Deficit} = \text{Requirement} - \text{Availability}$, and all three were included as features).
- **Feature Benefit:** Similar to the classification models (Naive Bayes and Logistic Regression), the inclusion of **seasonality features did not improve** the model, as the Base model already achieved perfect performance.

Overall Conclusion The Linear Regression model is an **ideal predictor** for this target variable. The R^2 of 1.0 and RMSE of 0.0 are unusual for real-world data and suggest a direct, deterministic, and perfectly linear relationship between the independent and dependent

variables. The **Base Linear Regression model** is the most efficient solution due to its perfect performance with fewer features.

7 Machine Learning Model Evaluation Summary

This project evaluated four different machine learning algorithms—three classification and one regression—to analyze the relationship between spatial/temporal features and energy metrics.

Model Performance Metrics

The table below presents the performance metrics for both the base feature set and the extended feature set (including seasonality).

Model	Feature Set	Metric	Value	Interpretation
Linear Regression	Base	R^2	1.0	Perfect Fit (RMSE = 0.0); deterministic linear relationship.
Linear Regression	Base	RMSE	0.0	—
Linear Regression	Seasonality	R^2	1.0	Seasonality adds no impact.
Linear Regression	Seasonality	RMSE	0.0	—
Logistic Regression	Base	Accuracy	1.0	Perfect classification; classes are linearly separable.
Logistic Regression	Seasonality	Accuracy	1.0	Seasonality adds no value.
Naive Bayes	Base	Accuracy	1.0	Excellent performance due to highly separable classes.
Naive Bayes	Seasonality	Accuracy	1.0	Seasonality adds no value.
k-Nearest Neighbors	Base	Accuracy	0.829	Strong performance (82.9%), the least accurate classifier among the four.
k-Nearest Neighbors	Seasonality	Accuracy	0.829	Seasonality has no effect.

Table 2: Comparative Model Performance for Base and Seasonality Features

Overall Conclusion

- **Deterministic Data:** The near-perfect scores for Linear Regression, Logistic Regression, and Naive Bayes suggest that the target variable is highly deterministic and linearly predictable from the base features.
- **Feature Redundancy:** Seasonality features provide no additional predictive power, as the base models already achieve maximum or near-maximum performance.

8 Decision Analysis

The predictive framework developed in this project serves as a critical decision-support tool for stakeholders in the energy sector. By analyzing historical trends and building robust regression models, we are now able to simulate and forecast future energy requirements with high accuracy.

- **Energy Planning:** The model allows businesses and policymakers to forecast energy demand based on availability and location, enabling proactive energy planning.
- **Risk Mitigation:** Identifying energy gaps ahead of time helps avoid potential shortages and supports timely procurement or infrastructure adjustments.
- **Strategic Investment:** The analysis supports decisions on investing in renewable energy projects by showcasing the potential for sustainable energy planning.
- **Customized Forecasting Tool:** The implementation of an interactive prediction utility provides flexibility to test multiple scenarios and make localized decisions.

9 Research Work

9.1 Evaluation of Research on Energy Efficiency in Energy-Intensive Manufacturing Companies

- **Authors:** Tobias Knayer and Natalia Kryvinska
- **Published In:** Frontiers in Energy Research (2022)
- **Summary:** This study analyzes energy efficiency measures in energy-intensive manufacturing companies, highlighting gaps in research and implementation. A literature review of 101 studies was conducted using eight evaluation criteria, including company type, industry, and data collection methods. The research found that energy efficiency studies focus primarily on large manufacturing firms, with limited studies on SMEs and non-manufacturing sectors. A SWOT analysis was used to assess opportunities and barriers to energy efficiency adoption. The study suggests that policymakers should focus on the interactions between individual energy efficiency measures to optimize impact and drive effective policies.

9.2 A Study on Energy Consumption, Energy Saving, and Effectiveness of Alternate Energy Sources in India's Domestic Sector

- **Authors:** Anirban Nandy, Poulomi Chaki, O.P. Pandey
- **Published In:** International Journal of Research in Engineering and Technology (2016)
- **Summary:** This paper examines the relationship between electricity consumption and consumer income in India's domestic sector, emphasizing the need for alternative energy sources. It discusses the depletion of fossil fuel reserves and the growing potential of renewable energy sources, particularly solar power. A case study in Siliguri, West Bengal, was conducted to analyze consumer electricity expenses and the financial benefits of adopting renewable energy. The paper highlights government initiatives, such as subsidies for solar rooftop installations, and advocates for microgeneration technologies like solar PV panels and wind turbines to supplement domestic energy needs.

9.3 Renewable Energy in India: Current Status and Future Potentials

- **Authors:** Ashwani Kumar, Kapil Kumar, Naresh Kaushik, Satyawati Sharma, Saroj Mishra
- **Published In:** Renewable and Sustainable Energy Reviews (2010)

- **Summary:** This article provides a comprehensive overview of renewable energy sources in India, focusing on wind, solar, biomass, geothermal, and hydropower. It discusses India's energy challenges and the need to increase renewable energy adoption to meet future energy demands. The paper also highlights policy interventions, including the National Electricity Policy (2005) and Integrated Energy Policy (2006), aimed at promoting sustainable energy. The study suggests that India's renewable energy sector has immense potential but requires stronger governmental policies, financial incentives, and technological advancements to scale up adoption.

9.4 A Study on Energy Consumption, Energy Saving, and Effectiveness of Alternate Energy Sources in the Domestic Sector of India

- **Authors:** Anirban Nandy, Poulomi Chaki, O.P. Pandey
- **Published In:** International Journal of Research in Engineering and Technology (2016)
- **Summary:** This study examines the relationship between electricity consumption and consumer income in India's domestic sector. It highlights the growing energy demand and the need to transition from fossil fuels to renewable energy sources like solar and wind power. The paper presents a case study in Siliguri, West Bengal, demonstrating how solar photovoltaic systems can reduce electricity expenses and increase household savings. Additionally, the study discusses government incentives such as subsidies and loans for solar energy adoption. The findings suggest that renewable energy integration at the household level can significantly reduce the energy demand-supply gap and enhance economic sustainability.

9.5 Renewable Energy in India: Current Status and Future Potentials

- **Authors:** Ashwani Kumar, Kapil Kumar, Naresh Kaushik, Satyawati Sharma, Saroj Mishra
- **Published In:** Renewable and Sustainable Energy Reviews (2010)
- **Summary:** This article provides a comprehensive overview of renewable energy sources in India, focusing on wind, solar, biomass, geothermal, and hydropower. It discusses India's energy challenges and the need to increase renewable energy adoption to meet future energy demands. The paper also highlights policy interventions, including the National Electricity Policy (2005) and Integrated Energy Policy (2006), aimed at promoting sustainable energy. The study suggests that India's renewable energy sector has immense potential but requires stronger governmental policies, financial incentives, and technological advancements to scale up adoption.

10 Conclusion and Recommendations

10.1 Key Takeaways

- The project explored energy consumption, energy efficiency, and the role of renewable energy in energy-intensive sectors.
- A detailed analysis was performed using various regression models, including Linear Regression, Ridge Regression, Lasso Regression, and Random Forest.
- The results indicated that energy requirement predictions were highly accurate, with R^2 scores nearing 1 for all models, with Ridge Regression offering the best performance.
- Stationarity tests, normality tests, and other statistical methods confirmed the significance of the energy data, allowing for accurate modeling and forecasting.
- The project demonstrated the utility of machine learning techniques in forecasting energy requirements and availability.

10.2 Business Recommendations

- **Energy Efficiency Focus:** Businesses should integrate energy efficiency measures to reduce consumption and costs.
- **Investment in Renewable Energy:** Organizations should consider solar, wind, and other renewables for long-term savings and sustainability.
- **Policy Engagement:** Engage with policymakers to align strategies and leverage incentives for clean energy initiatives.
- **Data-Driven Decisions:** Use predictive modeling to optimize resource planning and prevent energy shortages.

10.3 Future Work

- Incorporate real-time data feeds for dynamic forecasting.
- Explore deep learning models (e.g., LSTM, GRU) for improved time series predictions.
- Extend the prediction model to include external factors such as weather, economic indicators, and policy changes.
- Deploy the model as a web-based decision-support dashboard for public or enterprise use.