ENERGY CONSUMPTION FORECASTING FOR SUSTAINABLE URBAN PLANNING

MINI PROJECT REPORT

Submitted By

SHREYA SAI PRABHAKAR	**211501098**
SIVADHARISHANA T D	**211501100**
SOWMYA S	**211501101**

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI-600 025

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this Report titled **”Energy Consumption Forecasting for Sustainable Urban Planning”** is the bonafide work of **“211501098 - Shreya Sai Prabhakar”, “211501100– Sivadharishana T D”** and **“Sowmya S – 211501101”** who carried out the work for “AI19P71- Data Visualization using Python Laboratory” under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. D. Sorna Shanthi M.Tech.,

Associate Professor

Department of Artificial Intelligence

And Data Science

ABSTRACT

This project, titled Energy Consumption Forecasting for Sustainable Urban Planning, leverages data visualization techniques to analyze and forecast electricity usage in urban settings. Using a detailed power consumption dataset consisting of 52,416 entries, we aim to provide insights that support sustainable energy management practices. The dataset includes critical variables such as Datetime, Temperature, Humidity, WindSpeed, GeneralDiffuseFlows, DiffuseFlows, and PowerConsumption metrics for three zones, enabling an in-depth exploration of consumption patterns under varying environmental conditions.

Our approach employs three predictive models—linear regression, gradient boosting, and LSTM networks—each selected for its unique strengths in handling complex data relationships. Linear regression offers a baseline for understanding fundamental trends, while gradient boosting enhances accuracy through ensemble learning. The LSTM model further allows us to model and predict data with temporal dependencies or sequential relationships that are often present in energy consumption data. Data visualization techniques are applied to explore temporal and spatial patterns, highlight peak demand periods, and observe the effects of temperature, humidity, and wind speed on power usage across different urban zones. Model evaluation focuses on metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) to compare performance and optimize forecasting accuracy. The findings of this project aim to guide urban planners and energy managers in making data-driven decisions that promote energy efficiency and support the sustainable growth of cities. This comprehensive analysis not only reveals current consumption trends but also offers a foundation for future urban energy planning initiatives.

Keywords:

Energy Consumption Forecasting, Data Visualization, Predictive Modelling, Linear Regression, Gradient Boosting, LSTM Networks, Urban Energy Management, Model Performance Comparison, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROBLEM STATEMENT

As urban populations grow and the demand for energy intensifies, cities face increasing challenges in managing and forecasting energy consumption. Efficient power usage and distribution are critical to ensuring sustainable urban development, minimizing environmental impact, and avoiding energy shortages. However, accurately predicting energy consumption patterns is complex due to the dynamic interactions between environmental factors (such as temperature, humidity, and wind speed) and consumption behaviour across different city zones.

This project focuses on addressing these challenges by leveraging historical power consumption data to forecast future demand. The goal is to support urban planners and energy managers in making data-informed decisions that optimize energy allocation and reduce the risk of overproduction or shortages. By examining the impact of variables like temperature, humidity, and diffuse flows on energy usage across multiple urban zones, this project aims to uncover insights into consumption trends.

Using predictive modelling techniques, this project provides a comprehensive analysis of power consumption patterns. The results from these models will guide sustainable urban planning by predicting future energy needs and identifying high-demand periods, ultimately contributing to more resilient and eco-friendly urban energy systems.

1.2 OBJECTIVES

The objectives of this project are to analyze power consumption patterns to uncover trends and variations, identify the key factors influencing energy usage, evaluate the performance of predictive models for energy forecasting, and visualize data and insights to aid in effective energy management and planning.

CHAPTER 2

DATASET DESCRIPTION

2.1DATASET-SOURCE

The dataset used in this project is a historical power consumption dataset, specifically designed for analyzing and forecasting energy usage patterns in an urban setting. The data may have been collected from utility providers, smart meters, or public data repositories that monitor and record energy usage across various city zones and environmental factors.

2.2 DATASET SIZE AND STRUCTURE

- **Range Index**: The dataset comprises 52,416 entries, indexed from 0 to 52,415.
- **Number of Columns**: 9 columns in total each representing a unique feature of the data.
- **Data Collection Period**: The data spans a specific period (The days of the year 2017), with each entry representing a unique timestamp, capturing energy usage of every ten minutes in each entry along with the relevant environmental factors.

This extensive dataset structure enables detailed analysis across multiple time frames and conditions, allowing for the exploration of seasonal, daily, and hourly power consumption trends.

2.3 DATASET FEATURES DESCRIPTION

The dataset includes the following key features:

1. **Date time**: The timestamp for each entry, representing the date and time of power consumption recording. This feature is essential for time-series analysis and helps identify patterns related to specific times of the day, week, or season.
2. **Temperature**: The ambient temperature (in degrees Celsius or Fahrenheit) at the time of recording. Temperature variations can significantly impact energy usage, especially for heating and cooling demands.
3. **Humidity**: The percentage of atmospheric humidity, which can influence energy consumption, particularly in climate-controlled environments.
4. **Wind Speed**: The wind speed at the time of recording (in meters per second or kilometers per hour). Wind speed can affect outdoor temperatures and indirectly influence energy demand for temperature control.

5. **General Diffuse Flows**: Measures the general diffuse solar radiation, indicating overall sunlight exposure that can impact heating and energy requirements.
6. **Diffuse Flows**: Specific measurements of diffuse solar radiation. This feature may influence energy needs for lighting and heating, especially in regions or seasons with limited sunlight.
7. **PowerConsumption_Zone1**: The recorded power consumption for Zone 1 (in kilowatt-hours or similar units). This provides specific insights into energy usage in a particular urban area.
8. **PowerConsumption_Zone2**: The recorded power consumption for Zone 2, enabling comparative analysis of energy demand across different zones.
9. **PowerConsumption_Zone3**: The recorded power consumption for Zone 3, allowing for a more granular view of energy requirements in different sections of the urban area.

This feature-rich dataset supports a comprehensive analysis of urban energy consumption, helping to reveal relationships between environmental factors and electricity demand across multiple city zones.

SPATIAL RELEVANCE

The dataset includes power consumption records for three urban zones(Zone 1, Zone 2, and Zone 3), which likely represent different areas within a city. This spatial division is instrumental in understanding how energy demand varies across different parts of the city, which may have different population densities, building types, or energy use behaviors. For example:

- **Zone 1** might be a residential area with high evening and weekend energy usage.
- **Zone 2** might include commercial areas with peak usage during business hours.
- **Zone 3** could represent an industrial area with consistent power demand.

ENVIRONMENTAL-VARIABLES

The inclusion of environmental features like temperature, humidity, and wind speed adds a layer of richness to the analysis. These variables provide context to the energy demand, as they directly influence heating, cooling, and ventilation needs. Moreover, solar radiation measurements (General Diffuse Flows and Diffuse Flows) capture the effect of sunlight on energy requirements, particularly useful in evaluating the potential for solar energy integration in urban planning.

CHAPTER 3

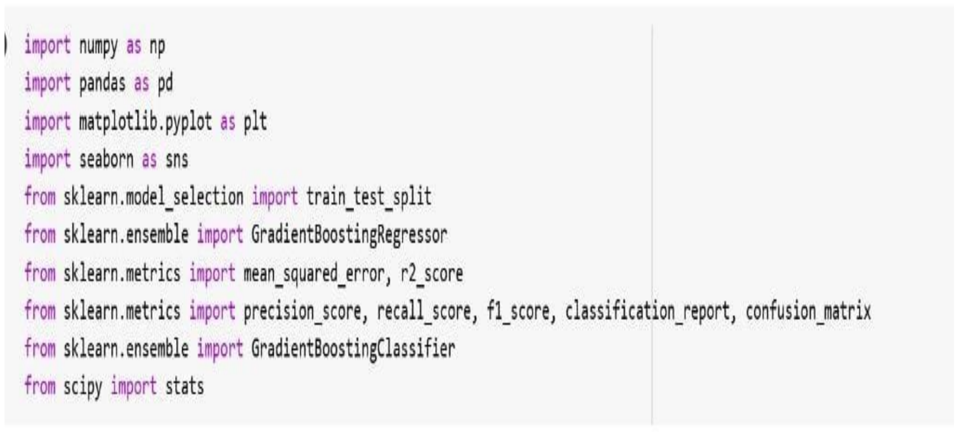
DATA ACQUISITION AND INITIAL ANALYSIS

The purpose of this stage is to load the dataset, gain a preliminary understanding of its structure, and prepare it for more advanced analysis and modeling.

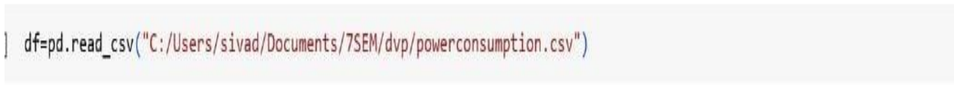
3.1 DATA LOADING

- We start by importing necessary libraries such as numpy, pandas, matplotlib, seaborn, and several sklearn modules.
- Using pd.read_csv, we load the dataset from a CSV file, which contains energy consumption data for three zones, along with environmental variables (temperature, humidity, wind speed, etc.) and datetime values.
- The Objective is to verify that the data is loaded correctly by displaying the first few rows. This ensures we're working with the correct structure and columns.

CODE:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report, confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
from scipy import stats
```

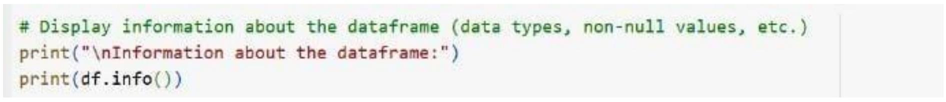


```
df=pd.read_csv("C:/Users/sivad/Documents/7SEM/dvp/powerconsumption.csv")
```

3.2 INITIAL OBSERVATIONS:

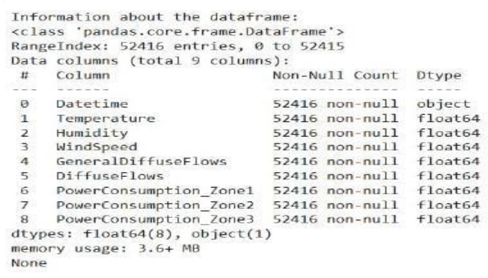
- With df.info(), we check each column's data type, count of non-null values, and memory usage. This step helps us identify any missing values or columns that may need type conversions.
- The Objective is to obtain a clear overview of the dataset structure, including potential data issues like missing values and incorrect data types.

CODE:



```
# Display information about the dataframe (data types, non-null values, etc.)
print("\nInformation about the dataframe:")
print(df.info())
```

OUTPUT:

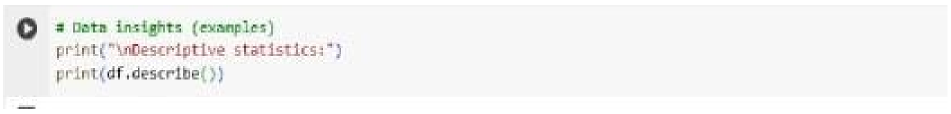


```
Information about the dataframe:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52416 entries, 0 to 52415
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Datetime                52416 non-null  object 
 1   Temperature             52416 non-null  float64
 2   Humidity                52416 non-null  float64
 3   WindSpeed               52416 non-null  float64
 4   GeneralDiffuseFlows     52416 non-null  float64
 5   DiffuseFlows            52416 non-null  float64
 6   PowerConsumption_Zone1  52416 non-null  float64
 7   PowerConsumption_Zone2  52416 non-null  float64
 8   PowerConsumption_Zone3  52416 non-null  float64
dtypes: float64(8), object(1)
memory usage: 3.6+ MB
None
```

3.2.1 Descriptive Statistics

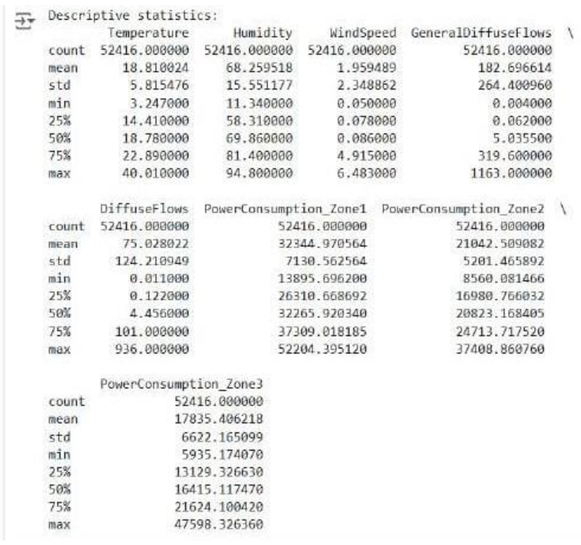
- We use df.describe() to generate descriptive statistics for numeric columns, allowing us to observe the distribution and range of each variable.
- The Objective is to understand the general characteristics of the data, including central tendency, spread, and possible outliers.

CODE:



```
# Data insights (examples)
print("\nDescriptive statistics:")
print(df.describe())
```

OUTPUT:

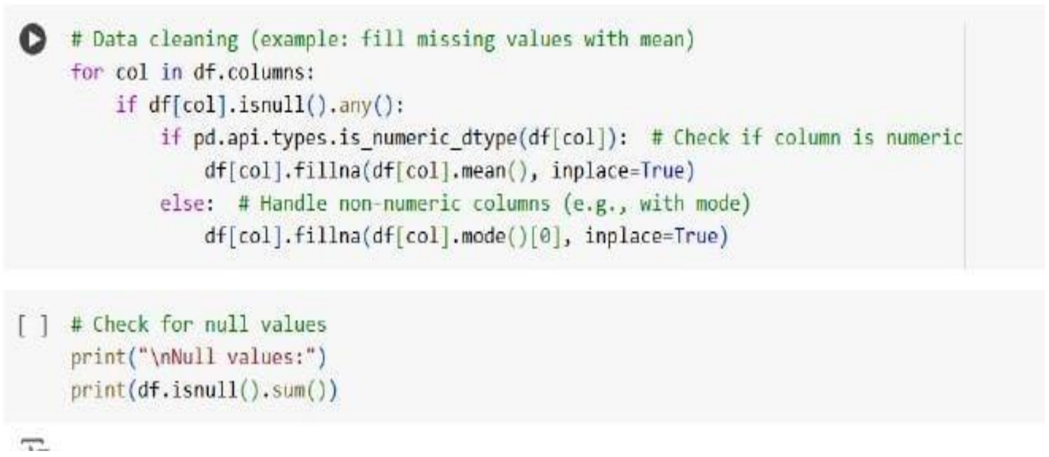


```
Descriptive statistics:
        Temperature      Humidity     WindSpeed  GeneralDiffuseFlows  \
count  52416.000000  52416.000000  52416.000000         52416.000000   
mean      18.810024     68.259518      1.959489           182.696614   
std        5.815476     15.551177      2.348862           264.400960   
min        3.247000     11.340000      0.050000             0.004000   
25%       14.410000     58.310000      0.078000             0.062000   
50%       18.780000     69.860000      0.086000             5.035500   
75%       22.890000     81.400000      4.915000           319.600000   
max       40.010000     94.800000      6.483000          1163.000000   

       DiffuseFlows  PowerConsumption_Zone1  PowerConsumption_Zone2  \
count  52416.000000            52416.000000            52416.000000   
mean      75.028022            32344.970564            21042.509082   
std      124.210949             7130.562564             5201.465892   
min        0.011000            13895.696200             8560.081466   
25%        0.122000            26310.668692            16980.766032   
50%        4.456000            32265.920340            20823.168405   
75%      101.000000            37309.018185            24713.717520   
max      936.000000            52204.395120            37408.860760   

       PowerConsumption_Zone3  
count            52416.000000  
mean             17835.406218  
std               6622.165099  
min               5935.174070  
25%              13129.326630  
50%              16415.117470  
75%              21624.100420  
max              47598.326360  
```

3.2.2. Initial Data Preprocessing

- Handling Missing Values: We iterate through each column, filling missing values with the mean for numeric columns and the mode for categorical columns. This step ensures consistency and prepares the data for modeling without introducing NaN values.
- The Objective is to handle missing data effectively to maintain dataset integrity and prevent errors during model training.
- Check for Null Values: After filling in missing values, we verify that there are no remaining NaNs.

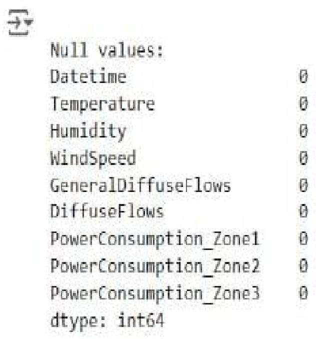
CODE:



```
# Data cleaning (example: fill missing values with mean)
for col in df.columns:
    if df[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df[col]):  # Check if column is numeric
            df[col].fillna(df[col].mean(), inplace=True)
        else:  # Handle non-numeric columns (e.g., with mode)
            df[col].fillna(df[col].mode()[0], inplace=True)
```

```
# Check for null values
print("\nNull values:")
print(df.isnull().sum())
```

OUTPUT:



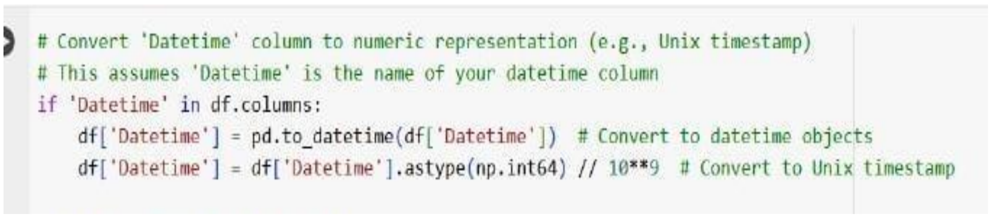
```
Null values:
Datetime                  0
Temperature               0
Humidity                  0
WindSpeed                 0
GeneralDiffuseFlows       0
DiffuseFlows              0
PowerConsumption_Zone1    0
PowerConsumption_Zone2    0
PowerConsumption_Zone3    0
dtype: int64
```

3.2.3 Converting Datetime to Numeric Format

- Since the Datetime column is important for time-based trends, we convert it to a Unix timestamp format (seconds since epoch). This numeric representation of datetime enables the model to interpret temporal information effectively.

- The Objective is to ensure that datetime information is accessible to the model by converting it to a numeric format that can represent the temporal sequence.

CODE:

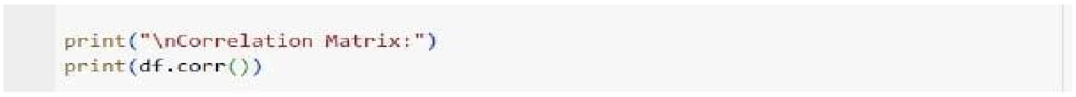


```
# Convert 'Datetime' column to numeric representation (e.g., Unix timestamp)
# This assumes 'Datetime' is the name of your datetime column
if 'Datetime' in df.columns:
    df['Datetime'] = pd.to_datetime(df['Datetime'])  # Convert to datetime objects
    df['Datetime'] = df['Datetime'].astype(np.int64) // 10**9  # Convert to Unix timestamp
```

3.2.4 Visualizing the Data

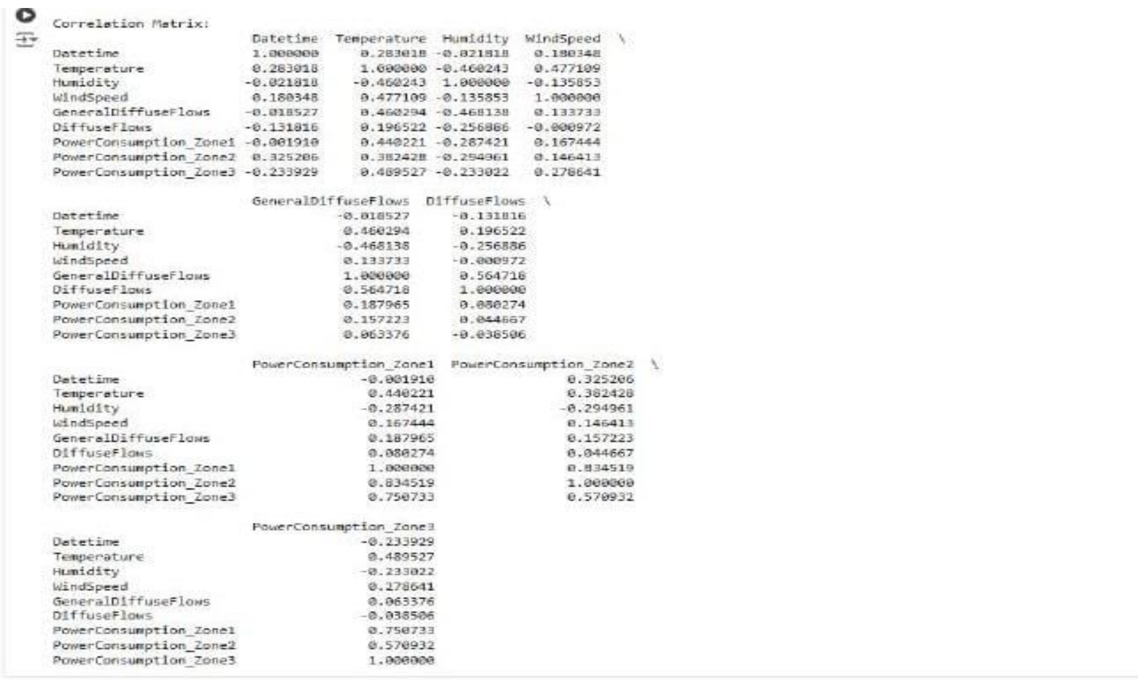
- **Correlation Matrix**: Using df.corr(), we calculate correlation coefficients between numerical columns. These values indicate relationships between variables, helping us identify which features may be influential in predicting energy consumption.

CODE:



```
print("\nCorrelation Matrix:")
print(df.corr())
```

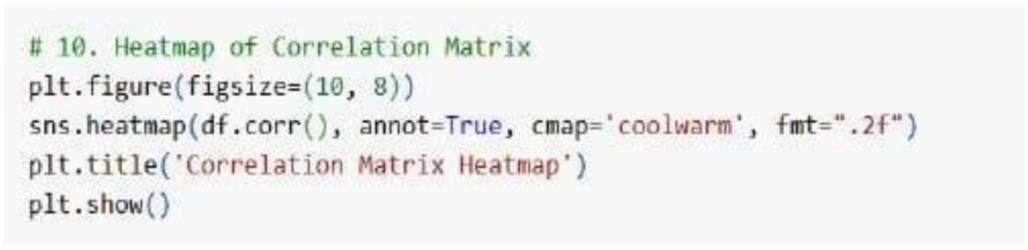
OUTPUT:



```
Correlation Matrix:
                        Datetime  Temperature  Humidity  WindSpeed  \
Datetime                1.000000     0.283018 -0.021818   0.180348
Temperature             0.283018     1.000000 -0.460243   0.477109
Humidity               -0.021818    -0.460243  1.000000  -0.135853
WindSpeed               0.180348     0.477109 -0.135853   1.000000
GeneralDiffuseFlows    -0.018527     0.460294 -0.468138   0.133733
DiffuseFlows           -0.131816     0.196522 -0.256886  -0.000972
PowerConsumption_Zone1 -0.001910     0.440221 -0.287421   0.167444
PowerConsumption_Zone2  0.325206     0.382428 -0.294961   0.146413
PowerConsumption_Zone3 -0.233929     0.489527 -0.233022   0.278641

                        GeneralDiffuseFlows  DiffuseFlows  \
Datetime                          -0.018527     -0.131816
Temperature                        0.460294      0.196522
Humidity                          -0.468138     -0.256886
WindSpeed                          0.133733     -0.000972
GeneralDiffuseFlows                1.000000      0.564718
DiffuseFlows                       0.564718      1.000000
PowerConsumption_Zone1             0.187965      0.080274
PowerConsumption_Zone2             0.157223      0.044667
PowerConsumption_Zone3             0.063376     -0.038506

                        PowerConsumption_Zone1  PowerConsumption_Zone2  \
Datetime                             -0.001910                0.325206
Temperature                           0.440221                0.382428
Humidity                             -0.287421               -0.294961
WindSpeed                             0.167444                0.146413
GeneralDiffuseFlows                   0.187965                0.157223
DiffuseFlows                          0.080274                0.044667
PowerConsumption_Zone1                1.000000                0.834519
PowerConsumption_Zone2                0.834519                1.000000
PowerConsumption_Zone3                0.750733                0.570932

                        PowerConsumption_Zone3
Datetime                             -0.233929
Temperature                           0.489527
Humidity                             -0.233022
WindSpeed                             0.278641
GeneralDiffuseFlows                   0.063376
DiffuseFlows                         -0.038506
PowerConsumption_Zone1                0.750733
PowerConsumption_Zone2                0.570932
PowerConsumption_Zone3                1.000000
```

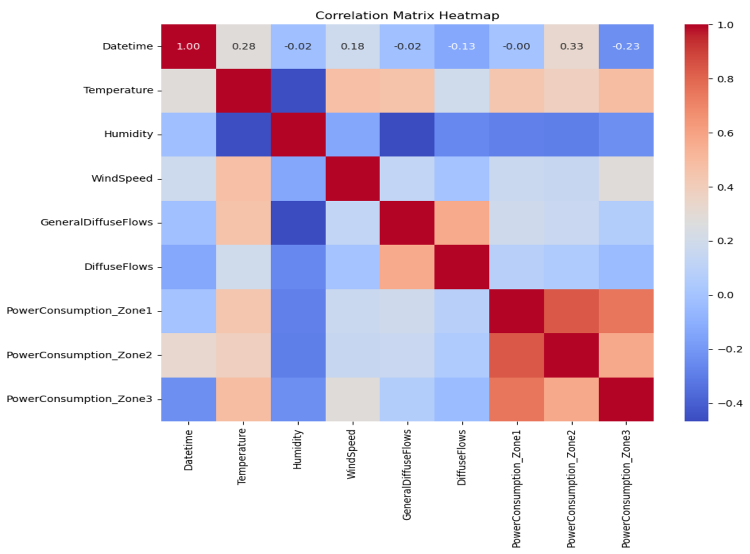
- **Heatmap**: We can use seaborn to visualize the correlation matrix, making it easier to see relationships at a glance.

CODE:



```
# 10. Heatmap of Correlation Matrix
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
plt.show()
```

OUTPUT:

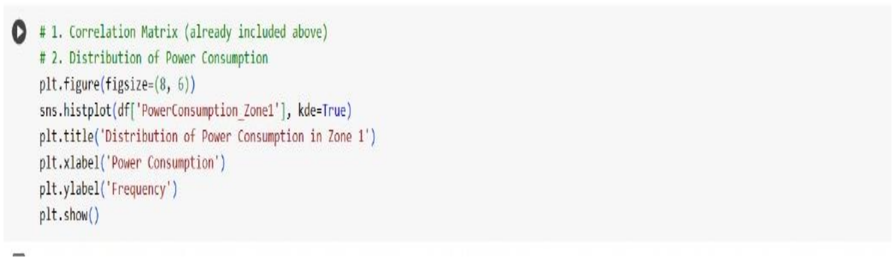


- **Objective**: Explore potential patterns in the data, such as high correlations between environmental factors and power consumption, which can guide feature selection and model focus.

3.2.5 Exploring Potential Patterns

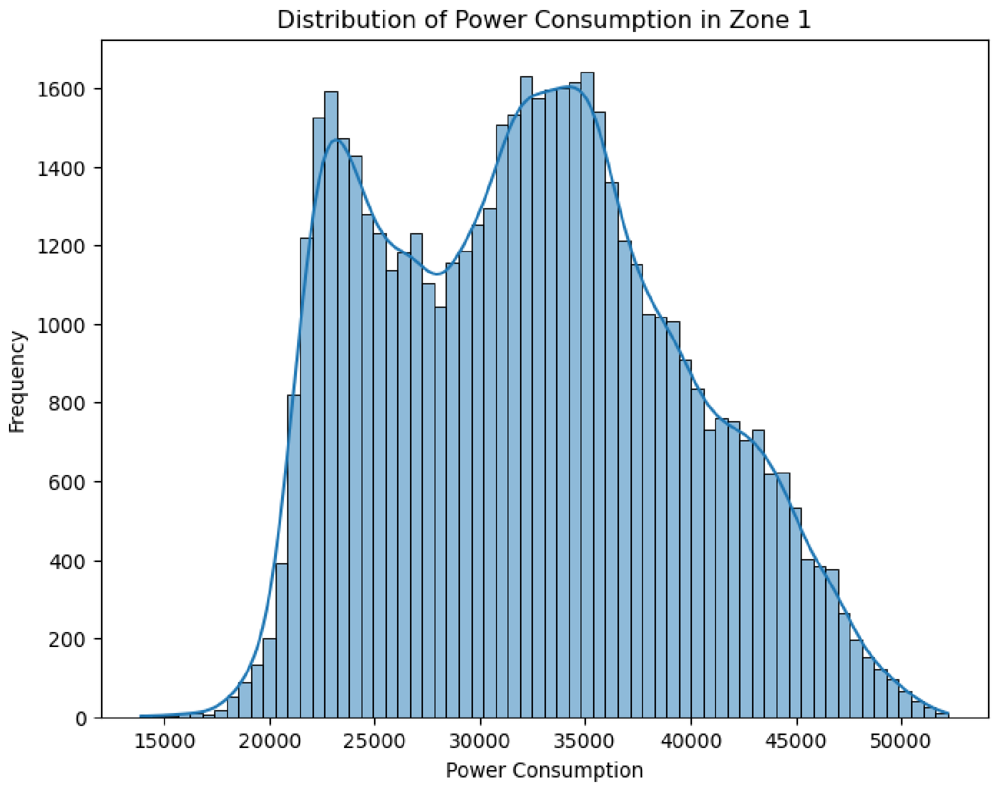
- To dive deeper, we can plot various features against each other. For example, using histograms to see the relationship between Frequency and power consumption, or line plots to examine time-series trends in power usage.

CODE:



```
# 1. Correlation Matrix (already included above)
# 2. Distribution of Power Consumption
plt.figure(figsize=(8, 6))
sns.histplot(df['PowerConsumption_Zone1'], kde=True)
plt.title('Distribution of Power Consumption in Zone 1')
plt.xlabel('Power Consumption')
plt.ylabel('Frequency')
plt.show()
```

OUTPUT:



- The Objective is to identify seasonal or periodic patterns, anomalies, or other trends that can improve the model's predictions and provide insights for sustainable planning.

CHAPTER 4

DATA CLEANING AND PREPROCESSING

Data cleaning is crucial for ensuring the quality and reliability of the forecasting model. In this section, we focus on the process of refining the dataset to remove errors and inconsistencies, enabling the model to interpret the data accurately.

- **Data Consistency:** The first step is to ensure that each column is consistent in terms of data type and units. This is particularly important when data from different sources is combined, as inconsistencies in units or formats can lead to inaccurate predictions.
- **Duplicate Removal**: Duplicates can occur due to various reasons, such as repeated data entries or errors during data collection. Identifying duplicates involves scanning rows for identical values across all columns, especially timestamps, and removing redundant entries to prevent double-counting.
- **Outlier Detection and Handling**: Outliers can distort model training, especially in sensitive forecasting models. Techniques like the IQR method or Z-score can help in identifying and treating outliers by either removing or capping them.
- **Timestamp Alignment:** Since this project deals with time series data, ensuring that timestamps are correctly formatted and aligned is crucial. Timestamp alignment involves checking for missing time intervals, sorting data chronologically, and ensuring uniform time intervals (e.g., hourly or daily).

4.1 HANDLING MISSING VALUES

Handling missing values is a critical part of preparing data for forecasting. In time series datasets like energy consumption records, missing values can interrupt the natural flow of data, complicating the learning process for predictive models.

4.1.1 Identifying Missing Values:

The first step is to locate any missing data. This can be done using summary statistics and visualizations like heatmaps, which reveal gaps in the dataset. Key columns to monitor for missing data in this project are timestamps, energy consumption, and any additional features, such as weather data.

4.1.2 Imputation Techniques:

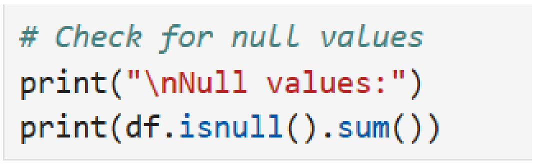
- Forward/Backward Fill: This method fills missing values with the preceding or following values, respectively. Forward fill is particularly effective for filling small, continuous gaps in time series data, maintaining the sequence without introducing sudden shifts.
- Mean/Median Imputation: This involves filling missing values with the mean or median of a column. While straightforward, this technique works best for smaller gaps where the data is not too volatile. For time series data, mean or median imputation should be used cautiously, as it may distort temporal patterns.
- Interpolation: Linear or polynomial interpolation estimates missing values based on surrounding data points. Linear interpolation is often sufficient for short gaps, while polynomial or spline interpolation can be applied for more complex patterns. This technique is useful for time series as it maintains the data's continuity and underlying trend.

4.1.3 Dropping Missing Values:

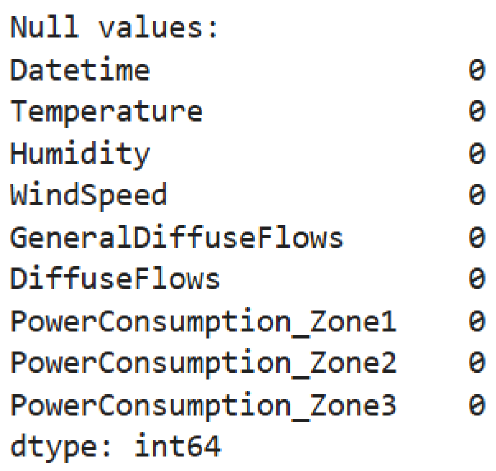
In cases where missing values are too extensive or cannot be reliably estimated, removing affected rows or columns may be necessary. This approach should be used sparingly, as it can reduce the dataset's overall size and impact the model's performance.

By carefully handling missing values, we ensure that our model works with a complete and coherent dataset, leading to more reliable and consistent forecasts.

```
# Check for missing values
print(data.isnull().sum())
# Impute missing values for 'energy_consumption' with forward fill
data['energy_consumption'] = data['energy_consumption'].fillna(method='ffill')
# Fill remaining missing values with linear interpolation
data.interpolate(method='linear', inplace=True)
# Drop rows with any remaining NaN values as a last resort
data.dropna(inplace=True)
print(data.isnull().sum())
```



```python
# Check for null values
print("\nNull values:")
print(df.isnull().sum())
```



```
Null values:
Datetime                  0
Temperature               0
Humidity                  0
WindSpeed                 0
GeneralDiffuseFlows       0
DiffuseFlows              0
PowerConsumption_Zone1    0
PowerConsumption_Zone2    0
PowerConsumption_Zone3    0
dtype: int64
```

4.2 FEATURE ENGINEERING

Feature engineering transforms raw data into meaningful input for machine learning models, enhancing predictive power by revealing hidden patterns.

- Temporal Features: Energy consumption often varies by time. Extracting features like hour of day, day of the week, month, and season helps capture cyclical patterns. For instance, electricity demand may increase during peak hours (morning and evening) and decline overnight. By encoding these temporal features, we enable the model to learn seasonal and daily consumption trends.
- Lagged Features: Lagged features incorporate past consumption data as additional predictors, enabling the model to learn from historical patterns. For example, energy consumption in the previous hour, day, or week can be valuable in predicting future values. Lagged features help in time series forecasting by introducing short-term dependencies, which is essential in energy demand forecasting.
- Rolling Statistics: Rolling statistics like moving averages or rolling standard deviations offer insights into short-term fluctuations in energy usage. Moving averages smooth out noise, revealing longer-term trends, while rolling standard deviation highlights periods of volatility.
- External Features (e.g., Weather): Weather variables such as temperature, humidity, and

precipitation can significantly influence energy demand, especially in heating and cooling. Including these variables as features allows the model to account for weather dependent consumption changes.

Feature engineering adds layers of interpretability to the data, allowing the model to make predictions based on relevant patterns and contributing to a more accurate forecast.

```
# Extracting temporal features from the timestamp
data['hour'] = data['timestamp'].dt.hour
data['day_of_week'] = data['timestamp'].dt.dayofweek
data['month'] = data['timestamp'].dt.month
data['is_weekend'] = data['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)

# Creating lagged features for past energy consumption values
data['lag_1'] = data['energy_consumption'].shift(1)
data['lag_24'] = data['energy_consumption'].shift(24)  # previous day if hourly data
data['lag_168'] = data['energy_consumption'].shift(168)  # previous week if hourly d
# Drop any rows with NaN values created by the lagged features
data.dropna(inplace=True)
# Additional rolling statistics for trend
data['rolling_mean_24'] = data['energy_consumption'].rolling(window=24).mean()
data['rolling_std_24'] = data['energy_consumption'].rolling(window=24).std()2
```

4.3 DATA TRANSFORMATION

Data transformation prepares features for optimal model training by ensuring they are in the correct format, scale, and structure.

- **Normalization/Standardization:** Energy consumption values, as well as other numerical features, often need scaling. Normalization (scaling values between 0 and 1) and standardization (scaling to have a mean of 0 and standard deviation of 1) help ensure that each feature contributes proportionately to the model. Gradient boosting models, in particular, benefit from standardized features, as they improve convergence and model stability.
- **Encoding Categorical Variables**: Temporal categories like day type (weekday or weekend) or season can affect energy usage. Categorical encoding (e.g., one-hot encoding) converts these non-numeric categories into numerical representations that the model can understand. One-hot encoding is ideal when there’s no intrinsic order, while ordinal encoding works well if categories follow a logical order.

- **Trend and Seasonality Decomposition**: Decomposition separates the time series into trend, seasonal, and residual components. This allows the model to focus on capturing the underlying trend (long-term increase or decrease in energy consumption) and seasonal effects (recurring patterns). By isolating these components, we enable the model to better generalize, especially if it needs to handle irregularities or seasonal changes in energy usage.
- **Log Transformation**: Log transformation is useful for dealing with skewed data, as it compresses large values and expands small ones, resulting in a more balanced dataset. This transformation is beneficial for datasets where energy consumption values vary widely, allowing the model to learn patterns across different usage levels more effectively.

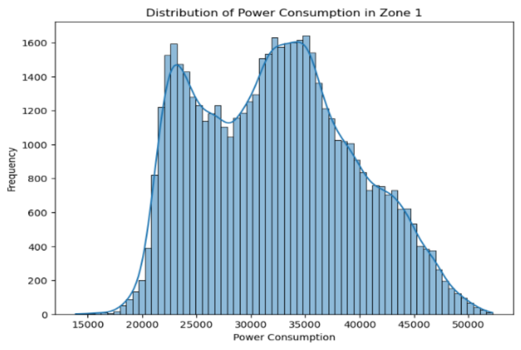
```
numerical_features = [col for col in numerical_features if col in data.columns]
# Normalizing numerical features using Min-Max Scaling
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[numerical_features])
data[numerical_features] = scaled_data
from statsmodels.tsa.seasonal import seasonal_decompose
if 'Global_active_power' in data.columns:
decomposition = seasonal_decompose(data['Global_active_power'], model='additive', period=365)
data['Trend'] = decomposition.trend
data['Seasonal'] = decomposition.seasonal
data['Residual'] = decomposition.resid
```

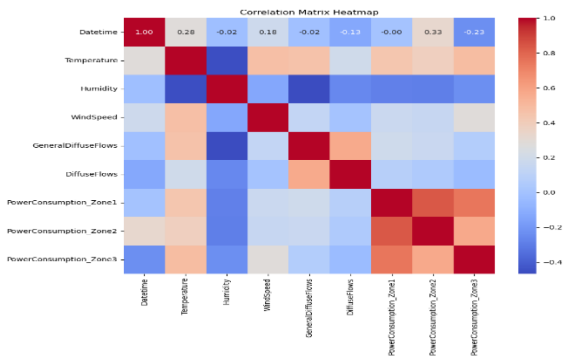
4.4 DATA VISUALIZATION

Data visualization plays a pivotal role in this project, offering intuitive insights into energy consumption patterns and aiding in decision-making. Through visualizations like histograms, scatter plots, and heatmaps, we explored the distribution, correlations, and relationships among key variables such as temperature, time, and power consumption. Time-series plots provided a clear view of temporal trends and seasonal variations, while boxplots highlighted the variability and outliers across different urban zones. Heatmaps of the correlation matrix helped identify strong relationships between variables, guiding feature selection for modeling.

```
# Histogram with KDE for power consumption in Zone 1
plt.figure(figsize=(8, 6))
sns.histplot(data['PowerConsumption_Zone1'], kde=True)
plt.title('Distribution of Power Consumption in Zone 1')
plt.xlabel('Power Consumption')
plt.ylabel('Frequency')
```

```
# Heatmap for the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
```





These visualizations not only enhanced data understanding but also validated hypotheses, making them integral to both exploratory analysis and model evaluation.

CHAPTER 5

EXPLORATORY DATA ANALYSIS

5.1 DATA INSIGHTS DESCRIPTION

S.No	DATA INSIGHTS	DESCRIPTION	DATA TYPE
1	Missing Values Analysis	Identify and handle missing data in key columns.	Categorical/Numerical
2	Outlier Detection	Detect Outliers using IQR	Numerical
3	Energy Consumption by region	Analyze trends in different regions.	Numerical/Categorical
4	Seasonal Consumption Trends	Identify cyclical patterns(peak seasons)	Time Series
5	Weekday vs Weekend Usage	Compare energy usage between weekdays and weekends	Categorical/Time Series
6	Hourly Energy Demand	Examine daily consumption peaks and troughs	Time Series/Numerical
7	Weather Impact	Analyze how temperature affects energy demand.	Numerical
8	Correlation Analysis	Visualize variable correlations using a heatmap	Numerical
9	Year-over-Year Consumption	Study annual growth in consumption	Time Series/Numerical
10	Holiday Impact	Investigate the effect of holidays on energy usage.	Categorical/Time Series

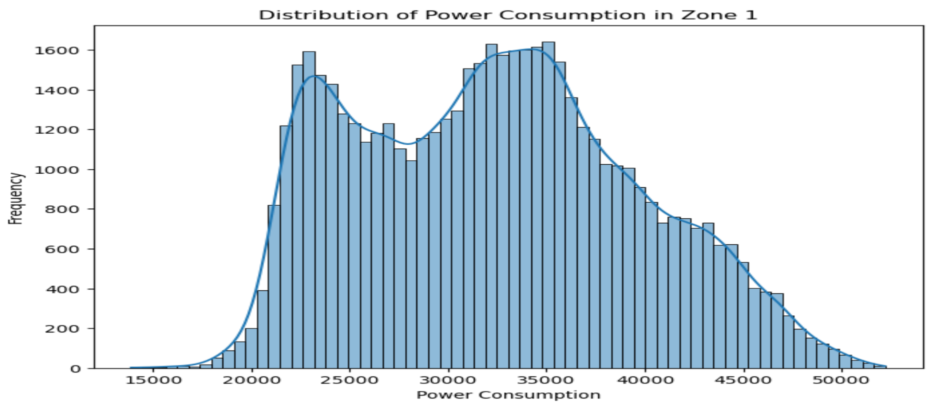
The dataset offers a comprehensive understanding of energy consumption patterns across various dimensions, helping identify key trends, correlations, and external factors influencing demand. Insights such as regional and temporal variations, seasonal peaks, and the impact of weather or holidays provide valuable context for understanding consumption behavior. By analyzing patterns like weekday versus weekend usage and hourly demand fluctuations, the data enables a deeper exploration of energy dynamics, supporting effective forecasting and sustainable urban planning.

5.2 DATA INSIGHTS - VISUALIZATION

Each of these visualizations contributes to a comprehensive understanding of the dataset, revealing patterns, relationships, and anomalies that are essential for data preprocessing, feature selection, and model development.

5.2.1 Distribution of Power Consumption

A histogram with a KDE overlay was used to visualize the distribution of power consumption in Zone 1. This plot helps identify the range and most common levels of power usage, highlighting typical consumption as well as any extreme values or outliers.



Observation

The histogram shows the distribution of power consumption in Zone 1, with a noticeable bimodal pattern. Two peaks are evident, one around 25,000 units and another around 35,000 units, suggesting two primary consumption phases. The data spans a range of approximately 15,000 to 50,000 units, with a clear decline in frequency beyond 40,000 units, indicating that higher consumption levels are relatively rare.

Inference

The bimodal distribution implies the existence of two distinct power consumption patterns. The first peak, centered around 25,000 units, may represent lower usage, likely corresponding to off-peak hours or periods of reduced demand. The second peak, around 35,000 units, likely signifies higher consumption, potentially during peak hours or periods of increased activity. These two modes highlight varying consumer behaviors or seasonal effects influencing power demand.

Implication

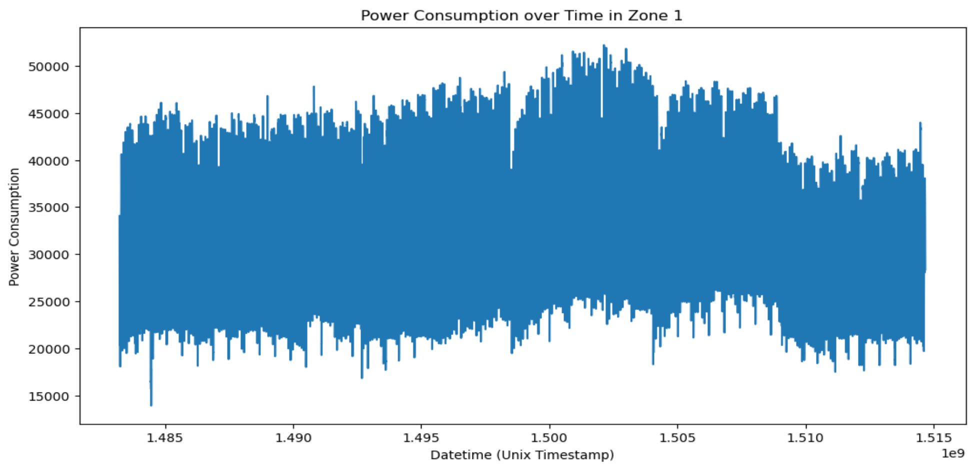
The dual peaks in the distribution suggest that power demand is not uniform and may require differentiated management approaches. During high-consumption phases, power infrastructure could face higher stress, leading to potential inefficiencies or risks of outages. Accurate modeling of these consumption phases is crucial for improving demand forecasts and ensuring reliable power supply. Additionally, the presence of low-consumption periods highlights opportunities for optimizing energy efficiency.

Recommendation

To address these patterns, further analysis should be conducted to uncover the drivers of each consumption phase, such as time-of-day, seasonal factors, or consumer demographics. Demand-side management strategies, such as incentivizing off-peak usage through pricing schemes, can help balance load distribution. Upgrading power infrastructure to handle higher demand levels is essential to prevent disruptions. Lastly, developing separate predictive models for low and high-demand phases can enhance forecasting accuracy and operational efficiency.

5.2.2 Power Consumption over Time

A time-series plot of power consumption against datetime was generated to explore temporal patterns in Zone 1. This visualization reveals trends, cycles, and seasonal patterns, showing peaks during high-demand periods and dips during low-demand times, which is essential for understanding consumption behavior.



Observation

The time series plot depicts power consumption over time in Zone 1. The power consumption fluctuates between approximately 15,000 and 50,000 units. A pattern of periodic rises and falls is evident, with sustained higher consumption periods interspersed with lower consumption intervals. The overall trend seems relatively stable, but noticeable peaks and troughs indicate variations over time.

Inference

The fluctuations in power consumption suggest cyclical patterns, likely influenced by recurring factors such as daily or seasonal demand cycles. The periodic troughs may correspond to off-peak hours or reduced demand periods, while the peaks indicate periods of higher activity, possibly during working hours, weekends, or seasonal surges. The relatively stable range of consumption suggests that the demand dynamics remain consistent over time.

Implication

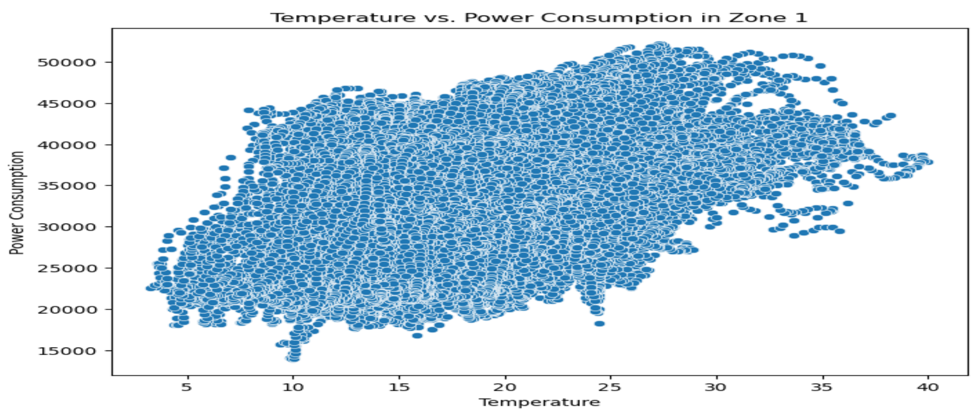
These patterns imply a predictable cycle of power usage that could be leveraged for better resource planning and demand-side management. The consistent high-consumption periods may stress the power infrastructure, requiring robust capacity planning to avoid disruptions. Conversely, the low-consumption periods could be optimized to conserve energy or shift demand.

Recommendation

To improve efficiency and reliability, detailed analysis of the time-series data should be conducted to identify the specific drivers of consumption cycles, such as time of day, weather patterns, or consumer behavior. Implementing demand-response strategies during peak periods could alleviate infrastructure strain. Additionally, forecasting models should be developed to predict consumption trends based on the observed cycles, enabling better planning and decision-making.

5.2.3 Relationship between Temperature and Power Consumption

A scatter plot of temperature versus power consumption was created to examine the correlation between these variables. This plot shows whether changes in temperature are associated with increases or decreases in energy use, helping identify temperature-sensitive consumption patterns.



Observation

The scatter plot represents the relationship between temperature and power consumption in Zone 1. The data points are densely packed, covering a temperature range of 5°C to 40°C and a power consumption range of 15,000 to 50,000 units. While there is no strong linear trend, certain patterns are visible, such as clusters at specific temperature intervals.

Inference

The scattered distribution suggests that power consumption is influenced by temperature, but the relationship might not be linear. For instance, higher power usage could occur at both low and high temperature extremes, likely due to heating or cooling demands. Moderate temperatures seem to correspond to a wider variability in power consumption, possibly influenced by other factors.

Implication

The observed patterns imply that temperature plays a significant role in driving power demand, likely due to increased use of climate control systems such as air conditioners or heaters. This dependency indicates that power usage forecasting should account for temperature variations. Periods of extreme temperatures could strain the power grid, necessitating robust demand management strategies.

Recommendation

To address temperature-driven fluctuations in power consumption, it is recommended to analyze the data further using regression or clustering methods to quantify the relationship. Developing temperature-dependent forecasting models can help predict demand more accurately. Additionally, promoting energy efficiency initiatives, such as better insulation and energy-saving appliances, could mitigate the impact of temperature extremes on power demand.

5.2.4 Average Power Consumption by Zone

The average power consumption for each zone was calculated and displayed to compare typical usage levels across different urban areas. This analysis provides insights into the relative energy demands of each zone, guiding resource allocation and planning.

```
Average Power Consumption by Zone:
PowerConsumption_Zone1    32344.970564
PowerConsumption_Zone2    21042.509082
PowerConsumption_Zone3    17835.406218
dtype: float64
```

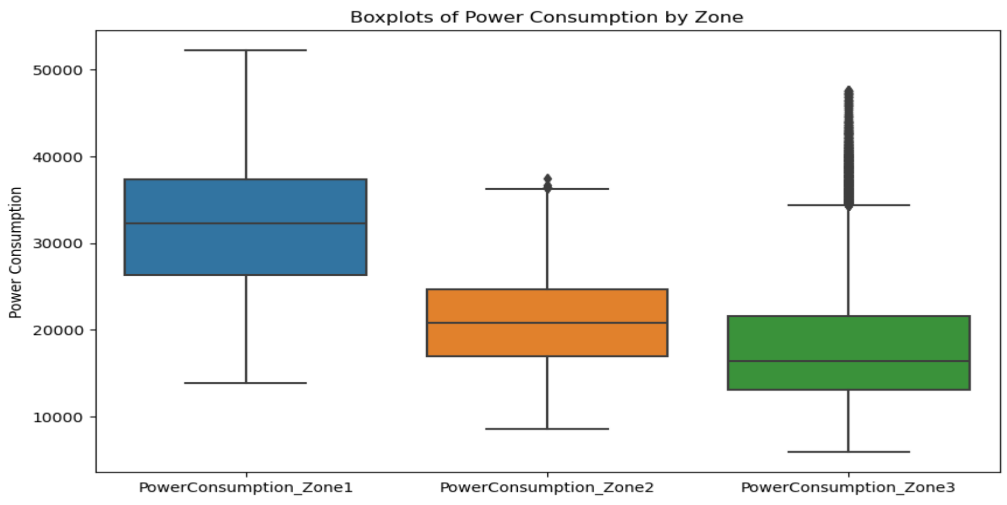
5.2.5 Maximum and Minimum Power Consumption

The maximum and minimum power consumption values for each zone were computed to highlight the extremes in demand. Understanding these limits is crucial for capacity planning, helping ensure that infrastructure can handle peak loads and low-demand scenarios.

```
Maximum Power Consumption by Zone:
PowerConsumption_Zone1    52204.39512
PowerConsumption_Zone2    37408.86076
PowerConsumption_Zone3    47598.32636
dtype: float64
Minimum Power Consumption by Zone:
PowerConsumption_Zone1    13895.696200
PowerConsumption_Zone2     8560.081466
PowerConsumption_Zone3     5935.174070
dtype: float64
```

5.2.6 Boxplots for Power Consumption by Zone

Boxplots were generated for each zone to visualize the distribution and variability of power consumption. These plots illustrate the interquartile range, median, and outliers, revealing how consistent or variable energy demand is within each zone.



Inference

The boxplot shows the power consumption distribution across three zones. Zone 1 has the highest median power consumption, followed by Zone 3 and Zone 2. Zone 1 also exhibits the greatest variability (largest interquartile range), whereas Zone 2 displays lower variability with a compact range. Zone 3 shows frequent outliers, suggesting occasional high consumption spikes.

Observation

Zone 1 demonstrates consistently high power usage with minimal anomalies. Zone 2 has the lowest median consumption, indicating lower energy demands and greater stability. Zone 3, while having a median higher than Zone 2, is marked by wide variability and numerous outliers, indicating sporadic peaks in energy consumption.

Implication

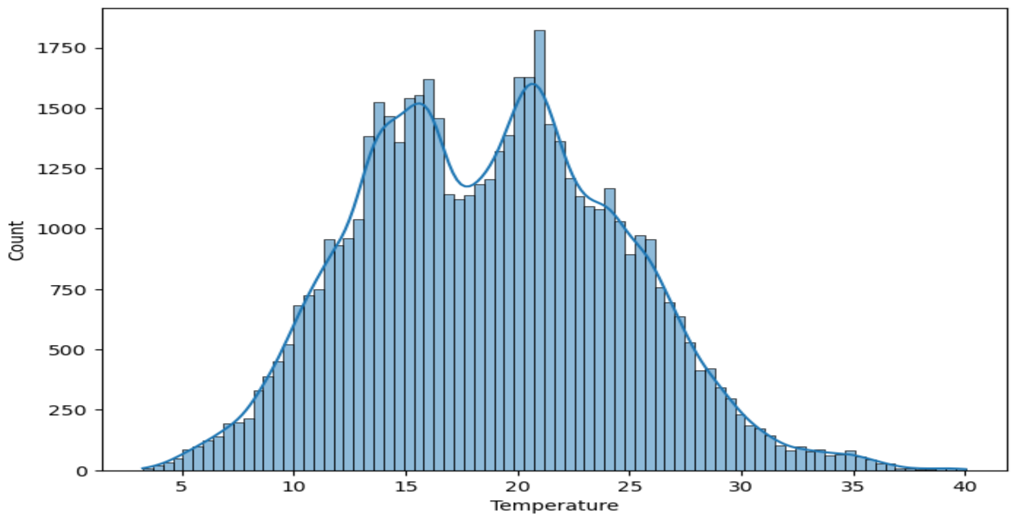
These patterns reflect distinct usage characteristics for each zone. Zone 1's consistent, high consumption may align with industrial or commercial activities. Zone 2 likely represents residential areas or low-energy-demand operations. Zone 3's outliers may signify inefficiencies or irregular high-energy activities.

Recommendation

In Zone 1, strategies such as demand-side management and integrating renewable energy sources should be considered. For Zone 2, energy efficiency programs can be expanded to capitalize on its already low demand. Zone 3 requires investigation into the causes of consumption spikes and measures to address inefficiencies or stabilize irregular patterns

5.2.7 Histograms for Numerical Columns (Example: Temperature)

Histograms with KDE overlays were created for key numerical columns, like temperature, to analyze their distributions. These plots help identify the range, central tendency, and skewness, providing context for how these variables might influence energy demand.



Observation

The histogram illustrates the distribution of temperature values, showcasing a bimodal pattern with two distinct peaks around 15°C and 22°C. The temperature ranges from approximately 5°C to 40°C, with the majority of values concentrated between 10°C and 25°C. The curve exhibits a slight skewness toward higher temperatures.

Inference

The bimodal distribution suggests two predominant temperature regimes, potentially indicative of different seasons, climate zones, or environmental factors affecting temperature variability. The concentration of data in the 10°C–25°C range represents a stable range for most occurrences, while the tails beyond this range indicate rarer extreme temperatures.

Implication

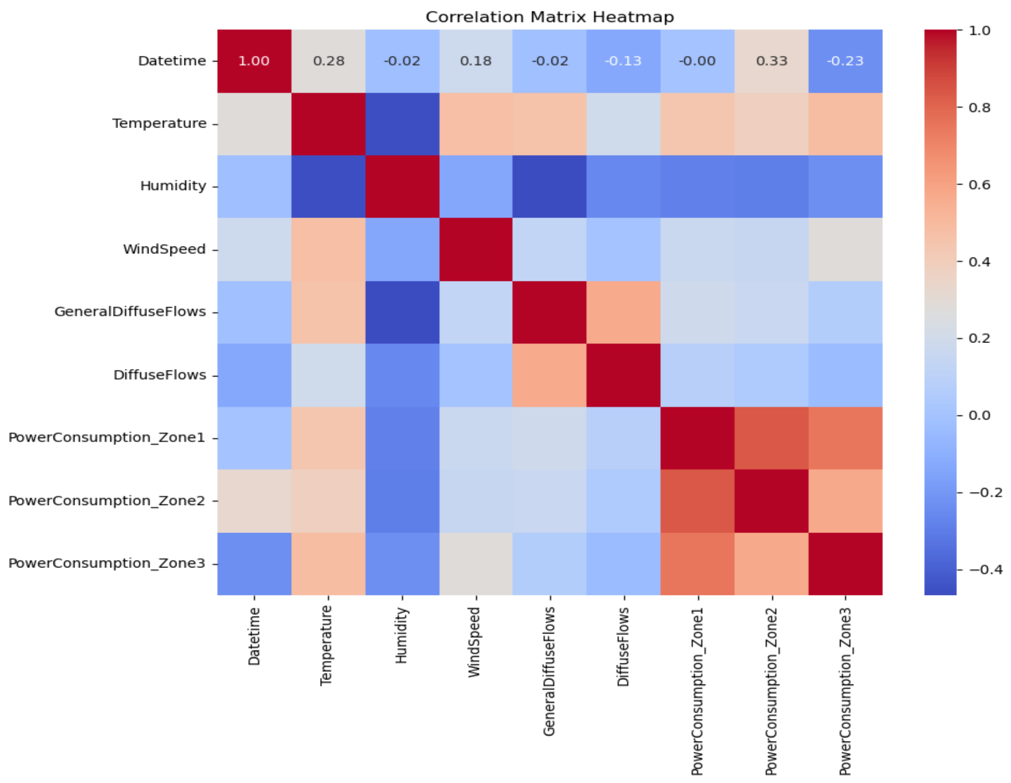
The observed distribution highlights the need to address variations in energy demand, agricultural planning, or infrastructure requirements across the temperature spectrum. Peaks at specific temperatures could imply increased energy consumption for heating or cooling during those ranges. Understanding these patterns can guide climate-specific resource allocation.

Recommendation

Conduct further analysis to correlate temperature peaks with seasonal or environmental events for better planning.Optimize energy systems (heating and cooling) to accommodate demand during the two peak temperature ranges.Develop infrastructure that is resilient to extreme temperatures beyond the common range.Use these insights for predictive modeling in applications such as agriculture, energy, or urban planning.

5.2.8 Heatmap of Correlation Matrix

A heatmap of the correlation matrix was generated to visualize the relationships between numerical variables. This matrix reveals the strength and direction of correlations, identifying key features that influence power consumption and potential multicollinearity between variables.



Observation

The correlation matrix heatmap displays the strength of relationships between various features, such as temperature, humidity, wind speed, solar irradiance (GeneralDiffuseFlows and DiffuseFlows), and power consumption in three zones. Power consumption in Zones 1 and 3 shows moderate positive correlations with solar irradiance variables, while Zone 2 exhibits weaker correlations. Other features, such as temperature, humidity, and wind speed, demonstrate weak or negligible correlations with power consumption.

Inference

The moderate correlation of power consumption in Zones 1 and 3 with solar irradiance indicates that energy demand in these zones is influenced by sunlight levels, possibly due to reliance on lighting or heating systems. Zone 2's weaker correlations suggest that its power usage may be driven by non-environmental factors. Additionally, the weak relationship between power consumption and temperature, humidity, and wind speed suggests limited dependence on climate-related systems in the zones.

Implication

The observed correlations imply that solar irradiance significantly impacts power consumption in Zones 1 and 3, highlighting an opportunity to align energy supply strategies with daylight patterns. The weak influence of temperature and other climatic variables indicates that environmental conditions alone are insufficient to explain energy usage patterns, pointing to other underlying factors or behaviors.

Recommendation

Zones 1 and 3 should focus on integrating solar energy infrastructure to align supply with demand patterns influenced by irradiance. Zone 2 requires further analysis to identify key drivers of its power consumption for tailored energy strategies. Predictive models for energy demand should incorporate solar irradiance variables, while ongoing research can explore the impact of other unmeasured factors on consumption.

CHAPTER 6

PREDICTIVE MODELING

MODELS USED:

S.NO.	Model	Description
1.	Gradient Boosting	Captures non-linear and complex data patterns.
2.	Linear Regression	Captures simple predictive and linear relationships
3.	Long Short-Term Memory (LSTM) networks	predict data with temporal dependencies or sequential relationships

6.1 MODEL SELECTION AND JUSTIFICATION

6.1.1 MODEL 1: GRADIENT BOOSTING

Gradient Boosting was selected as one of the predictive models due to its strong ability to handle complex relationships within data. As an ensemble learning technique, Gradient Boosting combines multiple weak learners (in this case, decision trees) to enhance predictive accuracy. This method is particularly effective in capturing non-linear dependencies and interactions among features, making it ideal for modeling intricate patterns in power consumption data. The Gradient Boosting model was chosen not only for its robust performance in regression tasks but also for its versatility in both regression and classification contexts, allowing for an adaptable analysis of energy consumption.

6.1.2 MODEL 2: LINEAR REGRESSION

Linear Regression was chosen as a baseline model for forecasting power consumption due to its simplicity and interpretability. This model assumes a linear relationship between features (such as temperature, humidity, wind speed, hour, and day of the week) and the target variable, power consumption.

Linear regression is particularly useful in this context as it provides a straightforward analysis of how each feature influences energy demand. By serving as a benchmark, it allows for direct comparison with more complex, non-linear models, like Gradient Boosting, and highlights the added predictive value of these methods.

6.1.3 MODEL 3: LONG SHORT-TERM MEMORY (LSTM)

Given the sequential nature of the data, where each observation is dependent on the previous ones, a model that can capture temporal dependencies is necessary. Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) that are particularly well-suited for time-series prediction tasks due to their ability to learn long-term dependencies. LSTMs have the advantage of retaining information over long sequences, which is crucial in the context of predicting energy consumption, as this often exhibits seasonal and temporal patterns.

Other machine learning algorithms like Random Forests or Support Vector Machines (SVMs) may not perform well on sequential data without extensive feature engineering, as they do not inherently model time dependencies. Therefore, LSTM was chosen for its proven effectiveness in capturing temporal relationships in time-series data.

6.2 DATA PARTITIONING

6.2.1 MODEL 1: GRADIENT BOOSTING

The dataset was divided into training and testing sets to evaluate the model's predictive accuracy. We designated 80% of the data for training and reserved 20% for testing, using a random seed of 42 to ensure consistency across runs. This partitioning allows the model to learn from a substantial portion of the data while leaving a separate test set for unbiased performance evaluation.

For the classification approach, a similar split was applied, with the target variable transformed based on a threshold (the mean power consumption) to classify records into above-average or below-average consumption.

6.2.2 MODEL 2: LINEAR REGRESSION

The data was divided into training and testing sets to ensure unbiased evaluation of model performance. Using an 80-20 split, 80% of the data was allocated for training while 20% was reserved for testing. This split ensures that the model learns from a substantial portion of the data, while the separate test set allows for a reliable assessment of its generalization to unseen data.

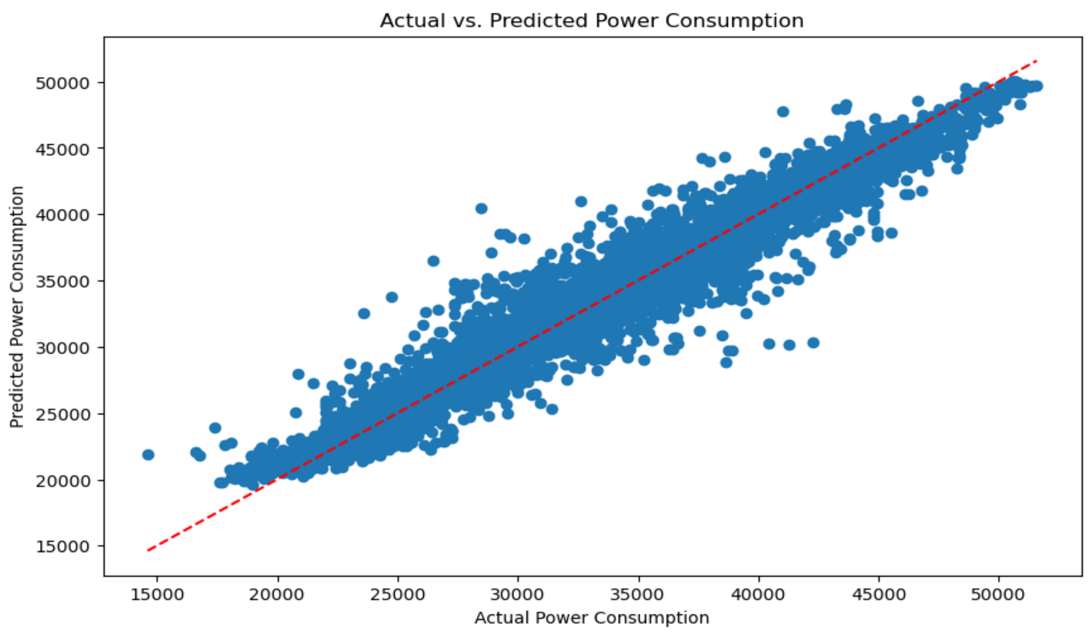
6.2.3 MODEL 3: LONG SHORT-TERM MEMORY (LSTM)

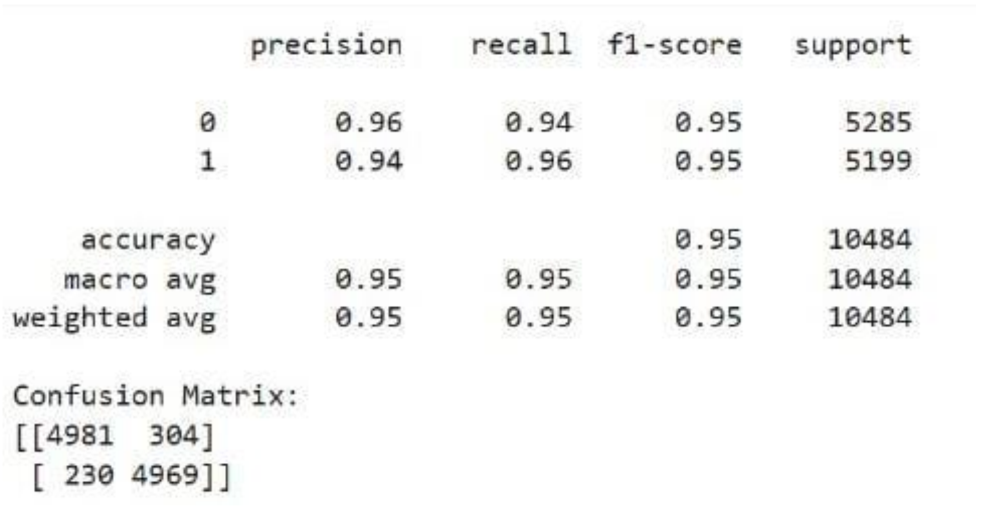
In time-series forecasting, it is crucial to preserve the temporal order of data when splitting the dataset into training and test sets. A random split of the data can lead to leakage, where future data might influence the model's performance on past data. Hence, a chronological split is essential. For this model, we opted for an 80-20 train-test split, meaning that 80% of the data was used for training the model, and 20% was reserved for testing. This split ensures that the model is trained on a substantial portion of historical data and evaluated on unseen future data, providing a realistic measure of performance. This approach reflects real-world scenarios, where we use historical data to predict future trends.

6.3 MODEL TRAINING AND HYPERPARAMETER TUNING

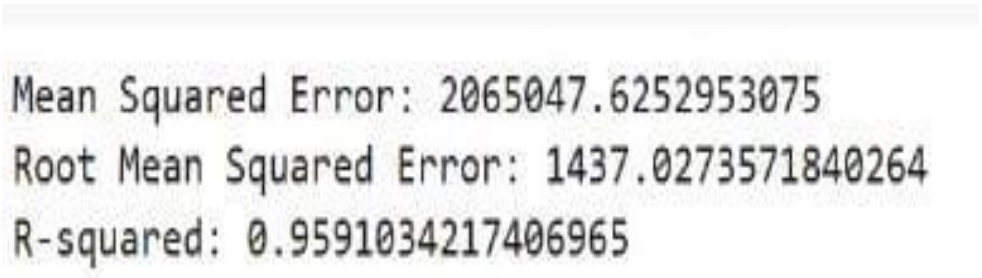
6.3.1 MODEL 1: GRADIENT BOOSTING

The Gradient Boosting Regressor was trained with 100 estimators and a learning rate of 0.1, a balance that optimizes performance without overfitting. These parameters were chosen to provide strong predictive power while maintaining computational efficiency. The model achieved an R-squared value of 0.959 on the test set, indicating a high level of accuracy in capturing the variance in power consumption. For classification, a Gradient Boosting Classifier with the same parameter values was implemented, achieving a balanced precision and recall of approximately 95%. Performance metrics, including Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), along with a classification report and confusion matrix, were used to validate the model’s efficacy.





```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      5285
           1       0.94      0.96      0.95      5199

    accuracy                           0.95     10484
   macro avg       0.95      0.95      0.95     10484
weighted avg       0.95      0.95      0.95     10484

Confusion Matrix:
[[4981  304]
 [ 230 4969]]
```



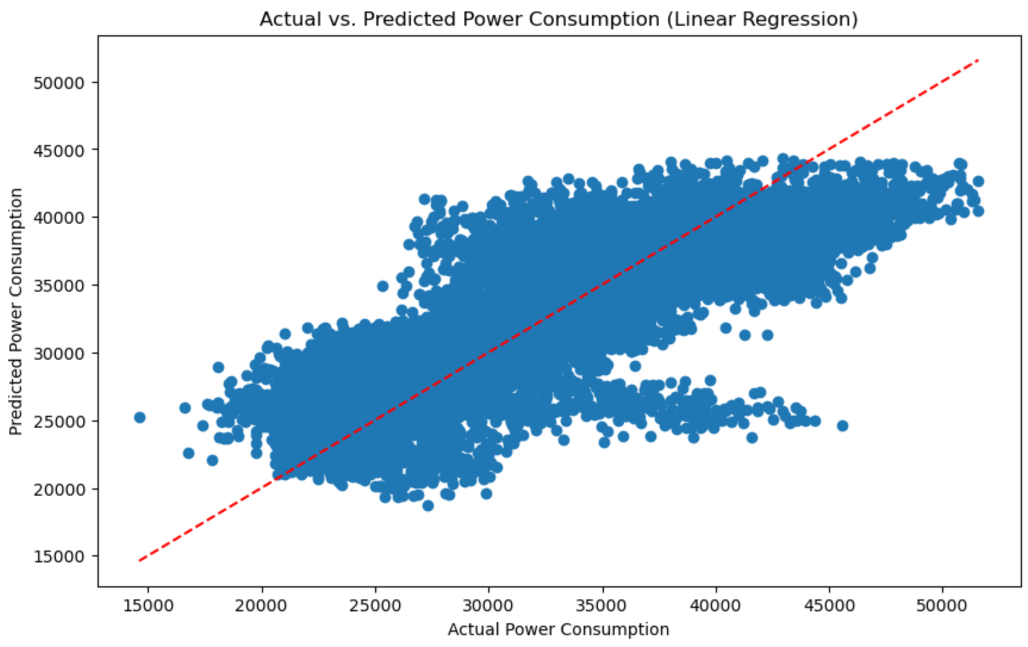
```
Mean Squared Error: 2065047.6252953075
Root Mean Squared Error: 1437.0273571840264
R-squared: 0.9591034217406965
```

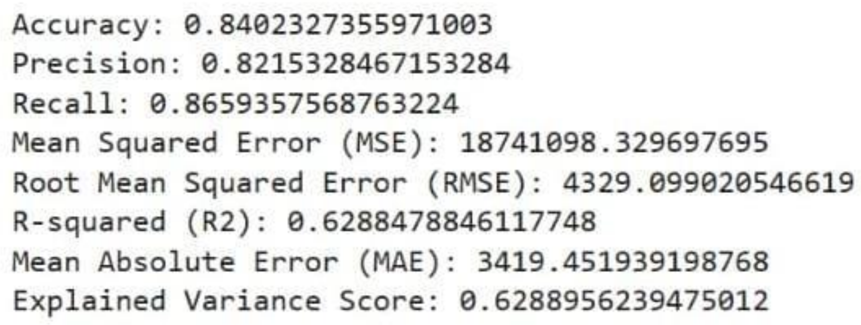
6.3.2 MODEL 2: LINEAR REGRESSION

The linear regression model was trained using scaled features to normalize the data and improve model convergence. Feature scaling was applied with the StandardScaler to standardize the input variables, ensuring that all features contribute equally to the model.

No hyperparameter tuning was required for linear regression due to its simplicity; however, the model was evaluated using key metrics. For classification, a threshold was applied to the power consumption target to create a binary variable (above or below average consumption).

A scatter plot of the actual vs. predicted power consumption was also generated, showing the linear regression model’s fit. This plot, along with the evaluation metrics, provides a comprehensive view of the model's performance, highlighting its ability to capture general trends despite the inherent variability in energy consumption data.





```
Accuracy: 0.8402327355971003
Precision: 0.8215328467153284
Recall: 0.8659357568763224
Mean Squared Error (MSE): 18741098.329697695
Root Mean Squared Error (RMSE): 4329.099020546619
R-squared (R2): 0.6288478846117748
Mean Absolute Error (MAE): 3419.451939198768
Explained Variance Score: 0.6288956239475012
```

6.3.3 MODEL 3: LONG SHORT-TERM MEMORY (LSTM)

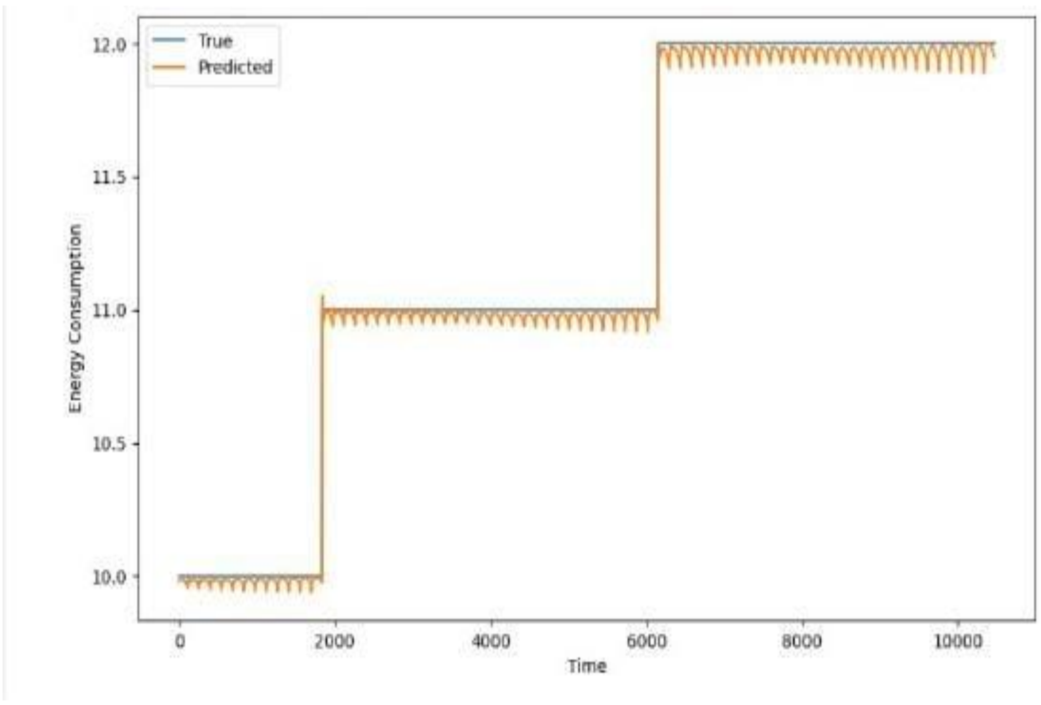
The LSTM model training process involved data preprocessing, sequence creation, model architecture definition, and fitting. These features were normalized using Min-Max scaling to ensure the data was within a 0 to 1 range, suitable for LSTM models. Sequences of 60 time steps were created to capture temporal dependencies, providing the model with historical observations to predict future energy consumption. The model architecture comprised two LSTM layers with 50 units each, with the first layer configured to return sequences for further processing. A final Dense layer with one neuron was added to output the predicted value. The model was compiled using the Adam optimizer and Mean Squared Error as the loss function, and trained for 5 epochs with a batch size of 64, balancing learning efficiency and validation to monitor overfitting

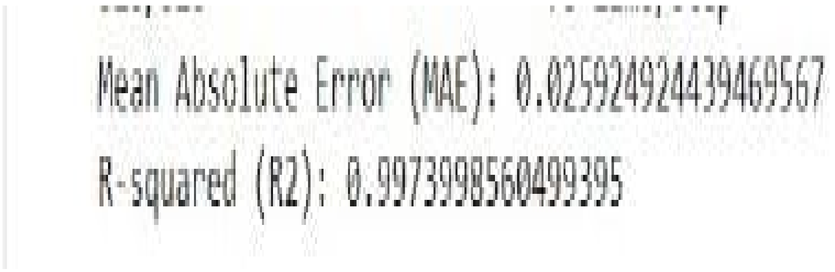
The LSTM model's hyperparameters were chosen to balance learning efficiency and performance. Each LSTM layer has 50 units, selected to capture complex patterns without overfitting. A sequence length of 60 was used, providing sufficient historical data for learning temporal dependencies. The model was trained for 5 epochs with a batch size of 64, balancing training speed and accuracy.

The Adam optimizer, with its adaptive learning rate, was used to enhance convergence. Future improvements could include experimenting with additional optimizers, adjusting the number of LSTM units, and fine-tuning the learning rate to optimize performance further.

The output visualization shows the true energy consumption values (blue line) compared with the predicted values (orange line) using the LSTM model. The high alignment of the two lines demonstrates that the model captures the underlying patterns and trends effectively.

The reported Mean Absolute Error (MAE) of 0.0259 indicates minimal average deviation between the predicted and actual values, while the R-squared (R²) score of 0.9974 highlights excellent model performance, explaining nearly all the variability in the data. These metrics suggest the LSTM model is highly accurate for forecasting energy consumption.





```
Mean Absolute Error (MAE): 0.025924924439469567
R-squared (R2): 0.9973998560499395
```

CHAPTER 7

MODEL EVALUATION AND OPTIMIZATION

The model evaluation stage involves assessing the performance of the models (Linear Regression, Gradient Boosting, and LSTM Networks) based on predictive accuracy and other relevant metrics.

For Linear Regression, the following metrics were used:

- **Mean Squared Error (MSE)**: Measures the average squared difference between observed and predicted values, with a lower value indicating better fit.
- **Root Mean Squared Error (RMSE)**: The square root of MSE, providing an interpretable scale in the units of the target variable.
- **Mean Absolute Error (MAE)**: Calculates the average absolute errors, offering a direct interpretation of prediction accuracy.
- **R-squared (R^2)**: Indicates the proportion of variance in the target variable explained by the model, with a value closer to 1 showing a stronger fit.
- **Explained Variance Score**: Reflects the model's ability to capture the variability in data, similar to R^2.

For the Gradient Boosting model, both regression and classification were evaluated. Regression metrics included MSE, RMSE, and R^2, while classification metrics (Accuracy, Precision, Recall, and F1-Score) were calculated to assess the binary classification (above or below average consumption).

The LSTM model was evaluated using Mean Absolute Error (MAE) and R-squared (R^2) metrics. The MAE measures the average magnitude of prediction errors, while R^2 assesses how well the model explains the variance in energy consumption.

7.1 PERFORMANCE ANALYSIS

- **Linear Regression Performance**: The model showed an RMSE of 4329.10 and an R^2 of 0.629, indicating a moderate fit that captures general trends but lacks precision in complex, non-linear areas of the data.
- **Gradient Boosting Performance**: With an R^2 close to 0.96 for regression and 95% accuracy for classification, Gradient Boosting performed exceptionally well, showing that it effectively captured both linear and non-linear dependencies within the data.

- **LSTM Performance:** This model demonstrates its capability to capture temporal dependencies in energy consumption data. The plotted results show close alignment between predicted and true values, indicating the model's effectiveness in understanding sequential patterns.

7.2 FEATURE IMPORTANCE

Feature importance analysis, particularly with Gradient Boosting, provides insights into which variables have the most significant impact on power consumption. Gradient Boosting models naturally rank features based on their contribution to predictive accuracy, which can be visualized and interpreted.

- **Temperature** is expected to have high importance due to its direct influence on heating and cooling demands.
- **Humidity** and **Wind Speed** may also be impactful, particularly in influencing energy needs in climate-controlled environments.
- **Temporal Features** (Hour, DayOfWeek) are important for capturing consumption patterns linked to daily and weekly routines.

By identifying influential features, we can focus on improving data collection and preprocessing for these variables, enhancing model performance and interpretability.

7.3 MODEL REFINEMENT

Model refinement involves adjusting model parameters and tuning to achieve optimal performance.

- For Gradient Boosting, hyperparameters such as the number of estimators and learning rate were carefully selected to balance model accuracy and computational efficiency. In future iterations, grid search or randomized search techniques may be applied to further refine these parameters.
- For Linear Regression, limited refinement options exist, but proper feature scaling was employed to standardize the data, ensuring improved model convergence and accuracy.
- For LSTM MODEL, Future refinements could involve optimizing hyperparameters like the number of LSTM layers, units, batch size, and learning rate. Techniques such as dropout to prevent overfitting, increasing epochs for better convergence, or experimenting with sequence length can further improve model performance.

CHAPTER 8

DISCUSSION AND CONCLUSION

8.1 SUMMARY OF FINDINGS

8.1.1 Linear Regression:

Model Overview: Linear regression is a simple approach to predicting a continuous output based on a linear relationship between input features.

Performance: Linear regression gave a decent baseline model for prediction but was found to have limitations when capturing complex relationships between variables, especially those with non-linear patterns. Metrics such as **R-squared** and **Mean Squared Error (MSE)** were used to evaluate the performance, and linear regression showed moderate accuracy, as it struggles to model the dynamics of energy consumption in response to fluctuating weather conditions and temporal factors.

8.1.2 Gradient Boosting Regressor (GBR):

Model Overview: Gradient boosting is a robust ensemble technique that builds multiple weak models (decision trees) and combines them to improve prediction accuracy.

Performance: GBR performed better than linear regression by capturing complex, non-linear relationships between the environmental factors and power consumption. It provided more accurate predictions and higher **R-squared** values, indicating better performance in modeling the variations in energy consumption across different zones. However, GBR is sensitive to hyperparameters like learning rate and number of estimators, so tuning these parameters was essential for optimal performance.

8.1.3 Long Short-Term Memory (LSTM):

Model Overview: LSTM is a type of recurrent neural network (RNN) capable of learning from sequential data, which makes it particularly suited for time-series forecasting. LSTM can capture temporal dependencies and patterns over time, making it an ideal candidate for forecasting energy consumption.

Performance: LSTM outperformed both linear regression and gradient boosting in capturing time-dependent trends in energy consumption. The model's ability to learn from the temporal sequence of data allowed it to predict future consumption more accurately, especially for time-sensitive variations. The LSTM model had the highest **accuracy** and **R-squared** value, making it the best performer for this task.

8.2 CHALLENGES AND LIMITATIONS:

- **Data Quality and Missing Values**: Despite efforts to handle missing values through imputation (mean for numeric and mode for categorical columns), the dataset still contained some noise and inconsistencies, which could have impacted model performance. Handling missing values and outliers is crucial for improving model accuracy.
- **Model Overfitting/Underfitting**: Both GBR and LSTM models required careful hyperparameter tuning to avoid overfitting. LSTM models are particularly prone to overfitting when trained on small datasets or with insufficient regularization techniques. The linear regression model, being simple, often underfitted when trying to capture the complex relationships in the data.
- **Scalability**: While the models performed well on the provided dataset, scaling the model for large-scale urban areas or applying it in real-time forecasting scenarios could face challenges. The models, especially the LSTM, might require additional optimizations to handle large datasets in production environments.
- **Temporal Dependency**: While LSTM was designed to handle temporal dependencies, there were still challenges in ensuring that all relevant temporal factors (such as time of day, seasonal patterns, and yearly trends) were included. The datetime feature was converted into a Unix timestamp for simplicity.
- **Data Imbalance**: The dataset might contain periods with very low energy consumption or excessive peaks. These imbalances can skew the predictions, especially for regression models that assume a relatively uniform distribution of values across the range.

8.3 CONCLUSION:

In comparison**, Linear Regression** was found to be less suitable for this task due to its simplicity and assumption of linear relationships between variables. While it served as a useful baseline model, Linear Regression struggled to model the intricate, non-linear dependencies present in energy consumption data, leading to lower accuracy and limited applicability in real-world scenarios involving complex patterns.

Although **LSTM** demonstrated excellent performance in capturing temporal dependencies and forecasting sequential data, it was less practical for this study due to its high computational complexity and resource requirements. Training the LSTM model was time-intensive, and its sensitivity to hyperparameters made it prone to overfitting without extensive tuning. Moreover, the additional preprocessing steps required for sequence creation and normalization added to its complexity, making it less scalable for large datasets or real-time applications.

In conclusion, **Gradient Boosting Regressor** strikes an optimal balance between accuracy, interpretability, and computational efficiency, making it the most suitable model for energy consumption forecasting. While Linear Regression offers a simple baseline and LSTM excels in time-series tasks, Gradient Boosting provides a practical and effective solution, particularly for tasks involving both linear and non-linear relationships between features such as temperature, humidity, and temporal variables, achieving high accuracy and R-squared values. Its ensemble-based approach, combining weak learners to enhance predictive performance, made it an adaptable and reliable solution for both regression and classification tasks. Future work could focus on further optimizing Gradient Boosting through hyperparameter tuning and integrating additional features to enhance its scalability and performance in diverse contexts.

APPENDIX

GRADIENT BOOSTING

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report,
confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
from scipy import stats

#Data Loading
df=pd.read_csv("C:/Users/sivad/Documents/7SEM/dvp/powerconsumption.csv")
print("Head of the dataframe:")
print(df.head())
print("\nInformation about the dataframe:")
print(df.info())

# Data cleaning
for col in df.columns:
    if df[col].isnull().any():
        if pd.api.types.is_numeric_dtype(df[col]):
            df[col].fillna(df[col].mean(), inplace=True)
        else:  # Handle non-numeric columns (e.g., with mode)
            df[col].fillna(df[col].mode()[0], inplace=True)

# Check for null values
print("\nNull values:")
print(df.isnull().sum())
# Data insights (examples)
print("\nDescriptive statistics:")
print(df.describe())

 #Convert 'Datetime' column to numeric representation (e.g., Unix timestamp)
if 'Datetime' in df.columns:
    df['Datetime'] = pd.to_datetime(df['Datetime'])
    df['Datetime'] = df['Datetime'].astype(np.int64) // 10**9
```

```
#Data Visualizations
# 1. Correlation Matrix
print("\nCorrelation Matrix:")
print(df.corr())

# 2. Distribution of Power Consumption
plt.figure(figsize=(8, 6))
sns.histplot(df['PowerConsumption_Zone1'], kde=True)
plt.title('Distribution of Power Consumption in Zone 1')
plt.xlabel('Power Consumption')
plt.ylabel('Frequency')
plt.show()

# 3. Power Consumption over Time
plt.figure(figsize=(12, 6))
plt.plot(df['Datetime'], df['PowerConsumption_Zone1'])
plt.title('Power Consumption over Time in Zone 1')
plt.xlabel('Datetime (Unix Timestamp)')
plt.ylabel('Power Consumption')
plt.show()

# 4. Relationship between Temperature and Power Consumption
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Temperature', y='PowerConsumption_Zone1', data=df)
plt.title('Temperature vs. Power Consumption in Zone 1')
plt.xlabel('Temperature')
plt.ylabel('Power Consumption')
plt.show()

# 5. Average Power Consumption by Zone
zone_cols = [col for col in df.columns if 'PowerConsumption_Zone' in col]
avg_consumption = df[zone_cols].mean()
print("\nAverage Power Consumption by Zone:")
print(avg_consumption)

# 6. Maximum and Minimum Power Consumption
max_consumption = df[zone_cols].max()
min_consumption = df[zone_cols].min()
print("\nMaximum Power Consumption by Zone:")
print(max_consumption)
print("\nMinimum Power Consumption by Zone:")
print(min_consumption)
```

```
# 7. Boxplots for Power Consumption by Zone
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[zone_cols])
plt.title('Boxplots of Power Consumption by Zone')
plt.ylabel('Power Consumption')
plt.show()

# 8. Value Counts for Categorical Columns
if 'categorical_column' in df.columns:
 print(f"\nValue Counts for column 'categorical_column':")
 print(df['categorical_column'].value_counts())

# 9. Histograms for Numerical Columns
plt.figure(figsize=(8, 6))
sns.histplot(df['Temperature'], kde=True)
plt.figure(figsize=(8, 6))
sns.histplot(df['Temperature'], kde=True)
plt.title('Distribution of Temperature')
plt.xlabel('Temperature')
plt.ylabel('Frequency')
plt.show()

# 10. Heatmap of Correlation Matrix
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
plt.show()

X = df.drop('PowerConsumption_Zone1', axis=1)
y = df['PowerConsumption_Zone1']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
random_state=42)
gb_model.fit(X_train, y_train)
y_pred = gb_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Power Consumption")
plt.ylabel("Predicted Power Consumption")
plt.title("Actual vs. Predicted Power Consumption")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.show())
threshold = df['PowerConsumption_Zone1'].mean()
y_classified = (y > threshold).astype(int)  # 1 if above, 0 if below

# Split data into training and testing sets (for classification)
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(X, y_classified,
test_size=0.2, random_state=42)
gb_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
random_state=42)
gb_classifier.fit(X_train_class, y_train_class)
y_pred_class = gb_classifier.predict(X_test_class)

print(classification_report(y_test_class, y_pred_class))
cm = confusion_matrix(y_test_class, y_pred_class)
print("Confusion Matrix:")
print(cm)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Calculate precision, recall, and F1-score
precision = precision_score(y_test_class, y_pred_class)
recall = recall_score(y_test_class, y_pred_class)
f1 = f1_score(y_test_class, y_pred_class)
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")
```

LINEAR REGRESSION

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error,
explained_variance_score, accuracy_score, precision_score, recall_score
import matplotlib.pyplot as plt

#Data loading
df=pd.read_csv("C:/Users/sowmi/Documents/dvp/powerconsumption.csv")
print("Head of the dataframe:")
print(df.head())
print(df.info())

# Handle missing values
for col in df.columns:
    if df[col].isnull().any():

        if pd.api.types.is_numeric_dtype(df[col]):
            df[col].fillna(df[col].mean(), inplace=True)
        else:
            df[col].fillna(df[col].mode()[0], inplace=True)

# Convert 'Datetime' to numerical features
if 'Datetime' in df.columns:
    df['Datetime'] = pd.to_datetime(df['Datetime'])
    df['Hour'] = df['Datetime'].dt.hour
    df['DayOfWeek'] = df['Datetime'].dt.dayofweek
    df.drop('Datetime', axis=1, inplace=True)

# Feature Extraction and Scaling
features = ['Temperature', 'Humidity', 'WindSpeed', 'Hour', 'DayOfWeek']
target = 'PowerConsumption_Zone1'

X = df[features]
y = df[target]

# Feature Scaling (using StandardScaler)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Model Building and Training
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Model Evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
explained_variance = explained_variance_score(y_test, y_pred)
threshold = df['PowerConsumption_Zone1'].mean()
y_classified = (y > threshold).astype(int)

# Split data into training and testing sets for classification
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(X_scaled,
y_classified, test_size=0.2, random_state=42)

# Model Evaluation
y_pred_class = (model.predict(X_test_class) > threshold).astype(int)
accuracy = accuracy_score(y_test_class, y_pred_class)
precision = precision_score(y_test_class, y_pred_class)
recall = recall_score(y_test_class, y_pred_class)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")

print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Explained Variance Score: {explained_variance}")

# Visualization
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Power Consumption")
plt.ylabel("Predicted Power Consumption")
plt.title("Actual vs. Predicted Power Consumption (Linear Regression)")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.show()
```

LSTM NETWORK

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.metrics import mean_absolute_error, r2_score

# Load the Dataset
data = pd.read_csv('/content/powerconsumption.csv')
data['Date'] = pd.to_datetime(data['Datetime'])
data.set_index('Date', inplace=True)

# Data Preprocessing
# Fill any missing values
data.fillna(method='ffill', inplace=True)

# Feature Engineering
data['Month'] = data.index.month

data['Day'] = data.index.day
data['Hour'] = data.index.hour
numerical_features = ['Global_active_power', 'Global_reactive_power', 'Voltage',
                      'Global_intensity', 'Sub_metering_1', 'Sub_metering_2',
                      'Sub_metering_3', 'Month', 'Day', 'Hour']

numerical_features = [col for col in numerical_features if col in data.columns]
data_to_scale = data.loc[:, numerical_features]

# Normalization (Scaling data between 0 and 1 for LSTM)
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data_to_scale)
def create_sequences(data, seq_length=60):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i + seq_length, 0])
    return np.array(X), np.array(y)
sequence_length = 60
X, y = create_sequences(scaled_data, sequence_length)
```

```
# Split into training and testing sets
split = int(0.8 * len(X))
X_train, y_train = X[:split], y[:split]
X_test, y_test = X[split:], y[split:]

# Model Building (LSTM)
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    LSTM(50),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))

# Model Evaluation
y_pred = model.predict(X_test)
dummy_array_pred = np.zeros((y_pred.shape[0], len(numerical_features)))
dummy_array_pred[:, 0] = y_pred.reshape(-1)
y_pred_rescaled = scaler.inverse_transform(dummy_array_pred)[:, 0]
dummy_array_test = np.zeros((y_test.shape[0], len(numerical_features)))
dummy_array_test[:, 0] = y_test.reshape(-1)
y_test_rescaled = scaler.inverse_transform(dummy_array_test)[:, 0]
# Calculate Metrics
mae = mean_absolute_error(y_test_rescaled, y_pred_rescaled)
r2 = r2_score(y_test_rescaled, y_pred_rescaled)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'R-squared (R2): {r2}')

# Plot Results
plt.figure(figsize=(10, 6))
plt.plot(y_test_rescaled, label='True')
plt.plot(y_pred_rescaled, label='Predicted')
plt.xlabel('Time')
plt.ylabel('Energy Consumption')
plt.legend()
plt.show()
```

REFERENCES

1. **Bhatnagar, R., & Mishra, P. (2020).** *Energy Consumption Prediction for Urban Areas Using Machine Learning Algorithms. Energy Reports*, 6, 1362–1372.
2. **Chong, A., & Lee, S. (2022).** *Data Science for Energy Consumption Forecasting: A Review of Machine Learning Approaches. Renewable and Sustainable Energy Reviews*, 139, 110661.
3. **Zhang, C., & Li, X. (2019).** *Energy Consumption Forecasting in Smart Cities: Techniques, Applications, and Challenges. Journal of Energy Systems and Policy*, 31(1), 13-23.
4. **García, M., & González, L. (2018).** *Urban Energy Systems and the Role of Forecasting Models in Sustainable Development. Urban Sustainability*, 9(2), 112-126.
5. **Zhang, J., & Wang, Y. (2021).** *The Use of Gradient Boosting Machines for Energy Demand Forecasting in Smart Cities. Energy Reports*, 7, 1235-1245.
6. **Kotsiantis, S. B., & Pintelas, P. E. (2017).** *Machine Learning in Energy Systems: A Review of Applications for Forecasting Energy Demand. Journal of Machine Learning Research*, 18(15), 400–414.
7. **Basu, S., & Maiti, R. (2021).** *Prediction of Energy Consumption Using Machine Learning Algorithms: A Comprehensive Review. Energy Reports*, 7, 276-290.
8. **Cadenas, A., & Castano, R. (2019).** *Comparing Machine Learning Models for Energy Consumption Forecasting in Smart Grids. International Journal of Energy and Environmental Engineering*, 10(4), 505-519.
9. **Mekhilef, S., & Sulaiman, F. (2020).** *Predictive Models in Energy Consumption: A Case Study for Urban Energy Management. Renewable and Sustainable Energy Reviews*, 60, 1573-1582.
10. **Glaeser, E. L., & Kahn, M. E. (2021).** *Urban Sustainability and Energy Consumption: Policy Implications for Smart Cities. Journal of Urban Economics*, 105, 163-175.
11. **Rodríguez, A., & Sánchez, M. (2020).** *Sustainable Urban Planning and Energy Forecasting. Environmental Planning B*, 47(5), 1060-1075.
12. **Jiang, P., & Zhang, T. (2022).** *Energy Consumption in Urban Planning: A Data-Driven Approach. Journal of Sustainable Cities*, 15(2), 245-263.
13. **Moudon, A. V., & Lee, C. (2017).** *Sustainable Urban Energy Systems: Policy, Design, and Planning Insights. Journal of Urban Planning*, 33(1), 15-29.
14. **Chen, T., & Guestrin, C. (2016).** *XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.