Project Name : Large Scale Wave Energy Farm :

Submitted by : Veenalakshmi P V

Date: 28-10-2024

*Table of Contents : *

1. Introduction

2. Goal Of the Project

3. Data Story

4. Data Preprocessing

5. Data Cleaning

6. Data transformation

7. Handling Outliers

8. Data Splitting

9. Data Visualisation

10. Feature Importance

11. Model Selection

12. Feature Selection

13. Pipeline

14. Model Evaluation and Predictions

15. Hyper parameter tuning

16. Saving the Model

17. Load the Model

Introduction :

This project has significance in renewable energy research, as accurately predicting power output can help in optimizing the design and operation of wave energy converters, leading to more efficient and sustainable energy production.

***Goal Of this Project : ***

The primary goal of this project is to analyze and compare power consumption patterns between Perth and Sydney across two datasets with varying regional granularity: Perth_49 vs. Sydney_49 and Perth_100 vs. Sydney_100. Using machine learning regression models (e.g., Random Forest, Linear Regression), the project aims to:

1.Predict Power Consumption:

Create predictive models to accurately estimate power usage based on features provided in the dataset.
Identify Key Factors: Use feature importance analysis to understand which factors most influence power consumption in each region.

2.Detect Patterns and Differences::

there are significant differences in power usage patterns between Perth and Sydney and assess whether these vary with the dataset's regional granularity (49 vs. 100 regions).

3.Evaluate Model Performance and Overfitting:

Test models on both training and testing sets to ensure generalizability and prevent overfitting, allowing the model to perform well on unseen data.

4.Provide Insights for Energy Management:

Derive insights that could potentially help energy providers or policymakers in planning and optimizing power distribution in Perth and Sydney.

2. Data Story :

Comparing Power Consumption in Perth and Sydney :

In this data story, Analyzing the power consumption data from two Australian cities: Perth and Sydney. The datasets, labeled Perth_49, Sydney_49, Perth_100, and Sydney_100, represent energy consumption measurements from these cities over time. Each dataset contains information on power usage across different time intervals or features. Our goal is to extract insights into power consumption patterns, detect anomalies or outliers, and highlight key features that contribute to variations in energy usage.

Objective:
The objective of this analysis is to:

Compare power consumption between Perth and Sydney.
Detect outliers in the data using the Z-score method.
Visualize patterns in power usage using time-series and distribution plots.
Identify key features contributing to power consumption through feature importance analysis.

Data Preprocessing Step:

Load The Data Set :

In [5]:

```
import pandas as pd
import numpy as np

perth_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
perth_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
sydney_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WE
sydney_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\W

print(f"\nPerth_49 : {perth_49.head()}, \n\nPerth_100 : {perth_100.head()}, \n\
```

```
Perth_49 :        X1     Y1      X2     Y2      X3     Y3      X4      Y4     X5
Y5  ...  \
0  600.0   0.0  546.16  37.50  489.79  74.88  432.47  112.05  650.0  0.0  ...
1  593.0  12.0  546.16  37.50  489.79  74.88  432.47  112.05  644.0  8.0  ...
2  593.0  12.0  546.16  37.50  489.79  74.88  432.47  112.05  644.0  8.0  ...
3  593.0  12.0  546.16  37.50  489.79  74.88  432.47  112.05  644.0  8.0  ...
4  200.0   0.0  146.17  37.53   89.76  74.93   32.40  112.18  400.0  0.0  ...

    Power42   Power43    Power44   Power45    Power46    Power47   Power48  \
0  88867.92  98844.30  101283.59  98934.63  101624.58  100915.03  99625.68
1  88896.55  98759.79  101346.07  98873.59  101629.01  100934.53  99606.13
2  88919.83  98746.68  101346.15  98875.57  101618.32  100941.00  99611.35
3  88855.14  98760.96  101338.59  98971.58  101632.28  100943.59  99589.25
4  88005.30  98630.24  100432.73  98803.01  101064.48  100948.38  99028.87

    Power49    qW  Total_Power
0  96704.34  0.87   4102461.43
1  96718.39  0.87   4103361.41
2  96719.14  0.87   4103680.44
3  96735.04  0.87   4105661.06
4  96286.71  0.79   3752648.77

[5 rows x 149 columns],

Perth_100 :         X1    Y1      X2     Y2      X3     Y3      X4      Y4
X5   Y5  \
0  1000.0  0.0  946.08  37.42  889.67  74.76  832.02  112.10  1250.0  1.0
1   800.0  0.0  746.04  37.38  689.81  74.79  632.43  111.97  1200.0  0.0
2   600.0  0.0  545.98  37.50  489.87  74.95  432.52  112.15   650.0  0.0
3   600.0  0.0  546.09  37.41  489.70  74.76  432.35  111.90   800.0  0.0
4   600.0  0.0  545.95  37.52  489.69  74.99  432.46  112.16   800.0  0.0

   ...   Power93    Power94    Power95    Power96   Power97    Power98  \
0  ...  98711.68  102872.82  100743.44   99259.87  98909.46  101388.37
1  ...  96351.38  102253.02  101744.20   99482.45  99304.59  101953.23
2  ...  96985.29  102472.80  101757.63   99429.45  98709.14  101312.44
3  ...  76823.85   88005.41   98779.92  100260.30  98744.25  101144.58
4  ...  98195.95  102874.16  100256.15   99221.12  98969.54  101389.77

     Power99  Power100    qW  Total_Power
0  101025.35  98676.66  0.75   7257985.04
1  100878.42  99508.49  0.74   7103374.61
2  100979.86  99024.16  0.76   7335380.64
3  100835.43  98915.38  0.75   7187769.87
4  100924.02  98796.00  0.75   7260222.61

[5 rows x 302 columns],

Sydney_49 :        X1    Y1      X2     Y2      X3     Y3      X4     Y4     X5
Y5  ...  \
0    1.0  1.0    0.00  70.00    1.00  140.0   50.00  198.0  401.0    1.0  ...
1  598.0  0.0  595.82  77.69  593.70  150.0  549.00  198.0  798.0    0.0  ...
2  198.0  0.0  197.46  75.19  192.94  150.0   87.64  198.0  398.0    0.0  ...
3  598.0  0.0  596.97  69.41  592.69  143.8  549.00  198.0  398.0  200.0  ...
4  198.0  0.0  197.18  79.83  192.97  150.0   89.53  198.0  398.0    0.0  ...

    Power42   Power43   Power44   Power45   Power46   Power47   Power48  \
0  71909.82  70674.49  70972.33  90957.03  90903.63  87876.82  79499.23
1  68757.68  70665.50  69963.48  84511.25  85691.70  85211.51  76678.20
2  73675.80  77808.44  73519.82  91436.35  88770.60  86632.78  77932.46
3  68947.21  71668.05  69380.67  85191.27  84453.12  85300.41  78573.25
```

```
4  78367.97  79075.06  74354.03  85254.75  86978.69  86951.65  77671.87

    Power49    qW  Total_Power
0  68880.39  0.78   4065416.61
1  76119.53  0.76   3951216.37
2  69343.12  0.78   4022640.78
3  72527.16  0.75   3879223.41
4  74901.38  0.77   3974691.24

[5 rows x 149 columns],

Sydney_100      X1    Y1      X2      Y2      X3      Y3      X4     Y4     X5
Y5  ...  \
0    1.0  1.0    1.00  51.00    1.00  101.00    1.00  151.0  398.0  0.0  ...
1  198.0  0.0  197.18  80.53  193.59  150.00   77.58  198.0  598.0  0.0  ...
2  198.0  0.0  197.07  76.64  192.74  155.74   84.67  198.0  798.0  0.0  ...
3    1.0  1.0    1.00  51.00    1.00  101.00    1.00  151.0  398.0  0.0  ...
4  198.0  0.0  197.46  75.07  197.18  149.14  149.00  198.0  598.0  0.0  ...

    Power93   Power94   Power95   Power96   Power97   Power98   Power99  \
0  74018.52  71727.79  67966.45  63101.26  88826.02  86531.44  83786.68
1  63702.46  67776.99  65133.52  63138.74  82852.91  83519.30  81973.65
2  55788.34  59593.98  60073.60  59198.12  63377.08  72078.85  77435.62
3  66961.48  65716.93  66637.89  62562.54  80858.08  82656.53  82171.28
4  51814.27  59556.86  68341.92  70731.90  64192.86  69757.10  75581.40

   Power100    qW  Total_Power
0  73514.19  0.69   7247491.41
1  71781.34  0.67   7119352.90
2  67457.26  0.68   7148342.69
3  71713.30  0.69   7317998.83
4  69741.63  0.65   6925096.49

[5 rows x 302 columns]
```

info()

In [7]:
```
sydney_100_data = pd.read_csv('C:\\Users\\Admin\\Desktop\\Large Scale Wave Ener
print("Sydney 100 Dataset Info:")
print(sydney_100_data.info())

# For Perth 49 dataset
perth_49_data = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy
print("\n\nPerth 49 Dataset Info:")
print(perth_49_data.info())

sydney_49_data = pd.read_csv('C:\\Users\\Admin\\Desktop\\Large Scale Wave Energ
print("\n\nSydney 49 Dataset Info:")
print(sydney_49_data.info())

perth_100_data = pd.read_csv('C:\\Users\Admin\\Desktop\\Large Scale Wave Energy
print("\n\nPerth 100 Dataset Info:")
print(perth_100_data.info())
```

```
Sydney 100 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2318 entries, 0 to 2317
Columns: 302 entries, X1 to Total_Power
dtypes: float64(302)
memory usage: 5.3 MB
None


Perth 49 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36043 entries, 0 to 36042
Columns: 149 entries, X1 to Total_Power
dtypes: float64(149)
memory usage: 41.0 MB
None


Sydney 49 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17964 entries, 0 to 17963
Columns: 149 entries, X1 to Total_Power
dtypes: float64(149)
memory usage: 20.4 MB
None


Perth 100 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7277 entries, 0 to 7276
Columns: 302 entries, X1 to Total_Power
dtypes: float64(302)
memory usage: 16.8 MB
None
```

Describe ()

In [8]:
```python
import pandas as pd
import numpy as np

perth_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
perth_100 = pd.read_csv("C:\\Users\Admin\\Desktop\\Large Scale Wave Energy\\WEC
sydney_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WE
sydney_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\W


# For Sydney 100 dataset
print("\n\nSydney 100 Dataset Summary:")
print(sydney_100_data.describe())

print("\n\nSydney 49 Dataset Summery:")
print(sydney_49_data.describe())

print("\n\nPerth 100 Dataset Summery:")
print(perth_100_data.describe())

print("\n\nPerth 49 Dataset Summery:")
print(perth_49_data.describe())
```

```
25%       50.000000   300.000000    50.000000   600.000000     0.000000
50%       74.820000   500.000000   100.000000   700.000000     0.080000
75%       74.960000   632.750000   112.150000   850.000000    50.000000
max      990.000000  1000.000000   990.000000  1000.000000   919.590000

       ...        Power42       Power43        Power44        Power45  \
count  ...   36043.000000   36043.00000   36043.000000   36043.000000
mean   ...   93678.772248   96530.68484   96666.293181   97007.214249
std    ...    7401.226140    6709.53446    7020.690028    4829.877255
min    ...   52516.130000   56391.97000   53877.360000   53050.330000
25%    ...   88177.210000   94648.08000   96932.520000   97612.350000
50%    ...   93694.540000   98729.91000   99269.310000   98857.150000
75%    ...  100997.520000  100622.52000  100282.360000   99156.130000
max    ...  110945.940000  109400.43000  114194.520000  106702.150000

             Power46        Power47        Power48        Power49  \
count   36043.000000   36043.000000   36043.000000   36043.000000
mean    98466.265281   98106.278501   97462.663041   96134.920454
std      4978.194259    4263.508074    3134.420742    3889.098339
min     55401.380000   63028.260000   61717.310000   47257.430000
```

Handle Missing Data

In [9]:
```python
# Checking for missing values in all datasets
print("Missing values in WEC_Sydney_49:\n", sydney_49_data.isnull().sum()) # Ch
print("Missing values in WEC_Sydney_100:\n", sydney_100_data.isnull().sum()) #
print("Missing values in WEC_Perth_49:\n", perth_49_data.isnull().sum()) # Chan
print("Missing values in WEC_Perth_100:\n", perth_100_data.isnull().sum()) # Ch
```

```
Missing values in WEC_Sydney_49:
 X1              0
Y1              0
X2              0
Y2              0
X3              0
               ..
Power47         0
Power48         0
Power49         0
qW              0
Total_Power     0
Length: 149, dtype: int64
Missing values in WEC_Sydney_100:
 X1              0
Y1              0
X2              0
Y2              0
X3              0
               ..
Power98         0
Power99         0
Power100        0
qW              0
Total_Power     0
Length: 302, dtype: int64
Missing values in WEC_Perth_49:
 X1              0
Y1              0
X2              0
Y2              0
X3              0
               ..
Power47         0
Power48         0
Power49         0
qW              0
Total_Power     0
Length: 149, dtype: int64
Missing values in WEC_Perth_100:
 X1              0
Y1              0
X2              0
Y2              0
X3              0
               ..
Power98         0
Power99         0
Power100        0
qW              0
Total_Power     0
Length: 302, dtype: int64
```

In [ ]:
```
This data set have no missing values
```

In [ ]:
```
The output shows the results of checking each dataset for missing values. Here’

WEC_Sydney_49 Dataset: :
There are 149 columns in total. No columns contain missing values, as each colu

WEC_Sydney_100 Dataset:
This dataset has 302 columns, which is consistent with the inclusion of additic

WEC_Perth_49 Dataset:
Like WEC_Sydney_49, this dataset has 149 columns. No missing values are present

WEC_Perth_100 Dataset:
This dataset has 302 columns, with Power1 to Power100 included. All columns hav

Key Takeaways No Missing Data: Across all four datasets (WEC_Sydney_49, WEC_Syd

Data Completeness: The absence of missing values ensures that can directly proc
```

Handle Duplicates :

- Check for duplicate rows using duplicated().
- Remove duplicates: drop_duplicates()

In [10]:
```python
# Check for duplicate rows in Perth and Sydney datasets
perth_49_duplicates = perth_49.duplicated()
sydney_49_duplicates = sydney_49.duplicated()
perth_100_duplicates = perth_100.duplicated()
sydney_100_duplicates = sydney_100.duplicated()

# Print the number of duplicates found
print(f"\nPerth 49 Duplicates: {perth_49_duplicates.sum()}")
print(f"\nSydney 49 Duplicates: {sydney_49_duplicates.sum()}")
print(f'\nPerth 100 Duplicates:{perth_100_duplicates.sum()}')
print(f'\nSydney 100 Duplicates:{sydney_100_duplicates.sum()}')
```

```
Perth 49 Duplicates: 25107

Sydney 49 Duplicates: 13148

Perth 100 Duplicates:4540

Sydney 100 Duplicates:1084
```

In [4]:
```python
import pandas as pd

perth_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
perth_100 = pd.read_csv("C:\\Users\Admin\\Desktop\\Large Scale Wave Energy\\WEC
sydney_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WE
sydney_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\W


# Remove duplicates from both Perth and Sydney datasets
perth_49_cleaned = perth_49.drop_duplicates()
sydney_49_cleaned = sydney_49.drop_duplicates()
perth_100_cleaned = perth_100.drop_duplicates()
sydney_100_cleaned = sydney_100.drop_duplicates()

# Check the shape to confirm the removal
print(f"Perth 49 Shape after removing duplicates: {perth_49_cleaned.shape}")
print(f"Sydney 49 Shape after removing duplicates: {sydney_49_cleaned.shape}")
print(f"Perth 100 shape after removing duplicates: {perth_100_cleaned.shape}")
print(f"sydney 100 shape after removing duplicates: {sydney_100_cleaned.shape}"
```

```
Perth 49 Shape after removing duplicates: (10936, 149)
Sydney 49 Shape after removing duplicates: (4816, 149)
Perth 100 shape after removing duplicates: (2737, 302)
sydney 100 shape after removing duplicates: (1234, 302)
```

Data Transformation**

o Encode categorical variables: Use pd.get_dummies() or LabelEncoder.
o Scale numerical features: Apply StandardScaler or MinMaxScaler.

Data transformation is an essential part of this project, aiming to prepare raw datasets for model training, analysis, and visualization. Here’s a structured approach to the transformations applied to the Perth and Sydney datasets in this power consumption comparison project

In [6]:
```python
import pandas as pd

# Load dataset from a CSV file into a DataFrame
perth_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
perth_100 = pd.read_csv("C:\\Users\Admin\\Desktop\\Large Scale Wave Energy\\WEC
sydney_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WE
sydney_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\W


perth_49_encoded = pd.get_dummies(perth_49, drop_first=True)  # drop_first=True
perth_100_encoded = pd.get_dummies(perth_100,drop_first=True)
sydney_100_encoded = pd.get_dummies(sydney_100,drop_first=True)
sydney_49_encoded = pd.get_dummies(sydney_49,drop_first=True)
```

In [ ]:

Scale Numerical Features :

Scaling ensures that all numerical features are on the same scale, improving model performance.

```
Using StandardScaler (standardization):

this code handles the scaling of numerical features in the specified datasets
to prepare them for analysis or modeling. The choice of scaling method
depends on the specific requirements of your analysis, such as whether to
bound the features to a specific range or normalize them based on their
distribution.
```

In [20]:

```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Load the datasets
perth_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
perth_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
sydney_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WE
sydney_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\W

# Initialize scalers
min_max_scaler = MinMaxScaler()
standard_scaler = StandardScaler()

# Scale features for Sydney_49 vs Perth_49
sydney_49_scaled = min_max_scaler.fit_transform(sydney_49.select_dtypes(include
perth_49_scaled = min_max_scaler.fit_transform(perth_49.select_dtypes(include=[

# Scale features for Sydney_100 vs Perth_100
sydney_100_scaled = standard_scaler.fit_transform(sydney_100.select_dtypes(incl
perth_100_scaled = standard_scaler.fit_transform(perth_100.select_dtypes(includ

# Convert scaled data back to DataFrame
sydney_49_scaled_df = pd.DataFrame(sydney_49_scaled, columns=sydney_49.columns)
perth_49_scaled_df = pd.DataFrame(perth_49_scaled, columns=perth_49.columns)
sydney_100_scaled_df = pd.DataFrame(sydney_100_scaled, columns=sydney_100.colum
perth_100_scaled_df = pd.DataFrame(perth_100_scaled, columns=perth_100.columns)

# Optional: Save the scaled dataframes to CSV
sydney_49_scaled_df.to_csv('sydney_49_scaled.csv', index=False)
perth_49_scaled_df.to_csv('perth_49_scaled.csv', index=False)
sydney_100_scaled_df.to_csv('sydney_100_scaled.csv', index=False)
perth_100_scaled_df.to_csv('perth_100_scaled.csv', index=False)
```

Handling Outliers :

In [21]:

```
from scipy import stats
import numpy as np

# Combine Perth and Sydney data

combined_data = pd.concat([perth_49,perth_100,sydney_49,sydney_100], axis=0)

# Calculate Z-scores for relevant features
z_scores = np.abs(stats.zscore(combined_data[['Power91', 'Power92', 'Power93',
outliers_z = np.where(z_scores > 3)  # Set threshold for outlier detection (Z >

# Output detected outliers
print(f"Outliers detected at these indices: {outliers_z}")

# Remove or handle outliers (removing in this case)
combined_data_no_outliers = combined_data[(z_scores < 3).all(axis=1)]
```

```
Outliers detected at these indices: (array([], dtype=int64), array([], dtype=i
nt64))
```

Split the data

In [22]:
```
import pandas as pd

# Load datasets
perth_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
perth_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WEC
sydney_49 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\WE
sydney_100 = pd.read_csv("C:\\Users\\Admin\\Desktop\\Large Scale Wave Energy\\W

# Function to split data with 80% training and 20% testing, preserving time ord
def time_based_split(data, target_column):
    X = data.drop(columns=[target_column])
    y = data[target_column]
    split_index = int(0.8 * len(data))
    X_train, X_test = X[:split_index], X[split_index:]
    y_train, y_test = y[:split_index], y[split_index:]
    return X_train, X_test, y_train, y_test

# Specify target columns (replace with actual target names if different)
target_column = 'Total_Power'

# Split each dataset
X_train_perth_49, X_test_perth_49, y_train_perth_49, y_test_perth_49 = time_bas
X_train_sydney_49, X_test_sydney_49, y_train_sydney_49, y_test_sydney_49 = time
X_train_perth_100, X_test_perth_100, y_train_perth_100, y_test_perth_100 = time
X_train_sydney_100, X_test_sydney_100, y_train_sydney_100, y_test_sydney_100 =

# Print shapes to confirm splits
print("Perth_49 Training set size:", X_train_perth_49.shape, y_train_perth_49.s
print("Perth_49 Testing set size:", X_test_perth_49.shape, y_test_perth_49.shap
print("Sydney_49 Training set size:", X_train_sydney_49.shape, y_train_sydney_4
print("Sydney_49 Testing set size:", X_test_sydney_49.shape, y_test_sydney_49.s
print("Perth_100 Training set size:", X_train_perth_100.shape, y_train_perth_10
print("Perth_100 Testing set size:", X_test_perth_100.shape, y_test_perth_100.s
print("Sydney_100 Training set size:", X_train_sydney_100.shape, y_train_sydney
print("Sydney_100 Testing set size:", X_test_sydney_100.shape, y_test_sydney_10
```

```
Perth_49 Training set size: (28834, 148) (28834,)
Perth_49 Testing set size: (7209, 148) (7209,)
Sydney_49 Training set size: (14371, 148) (14371,)
Sydney_49 Testing set size: (3593, 148) (3593,)
Perth_100 Training set size: (5821, 301) (5821,)
Perth_100 Testing set size: (1456, 301) (1456,)
Sydney_100 Training set size: (1854, 301) (1854,)
Sydney_100 Testing set size: (464, 301) (464,)
```

In [ ]:

In [24]:
```
# For Perth_49
X_perth = perth_49[['Power41', 'Power42', 'Power43', 'Power44', 'Power45']]  #
y_perth = perth_49['Total_Power']  # Target

# For Sydney_49
X_sydney = sydney_49[['Power41', 'Power42', 'Power43', 'Power44', 'Power45']]
y_sydney = sydney_49['Total_Power']  # Target
```

In [ ]:

In [25]:
```
from sklearn.model_selection import train_test_split

# Assuming 'Power100' is the target variable want to predict

# Define feature columns (X) and target column (y)
X = combined_data[['Power41', 'Power42', 'Power43', 'Power44', 'Power45']]  # F
y = combined_data['Power100']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# Check the shapes of the resulting datasets
print(f"Training set size: {X_train.shape}, Testing set size: {X_test.shape}")
```

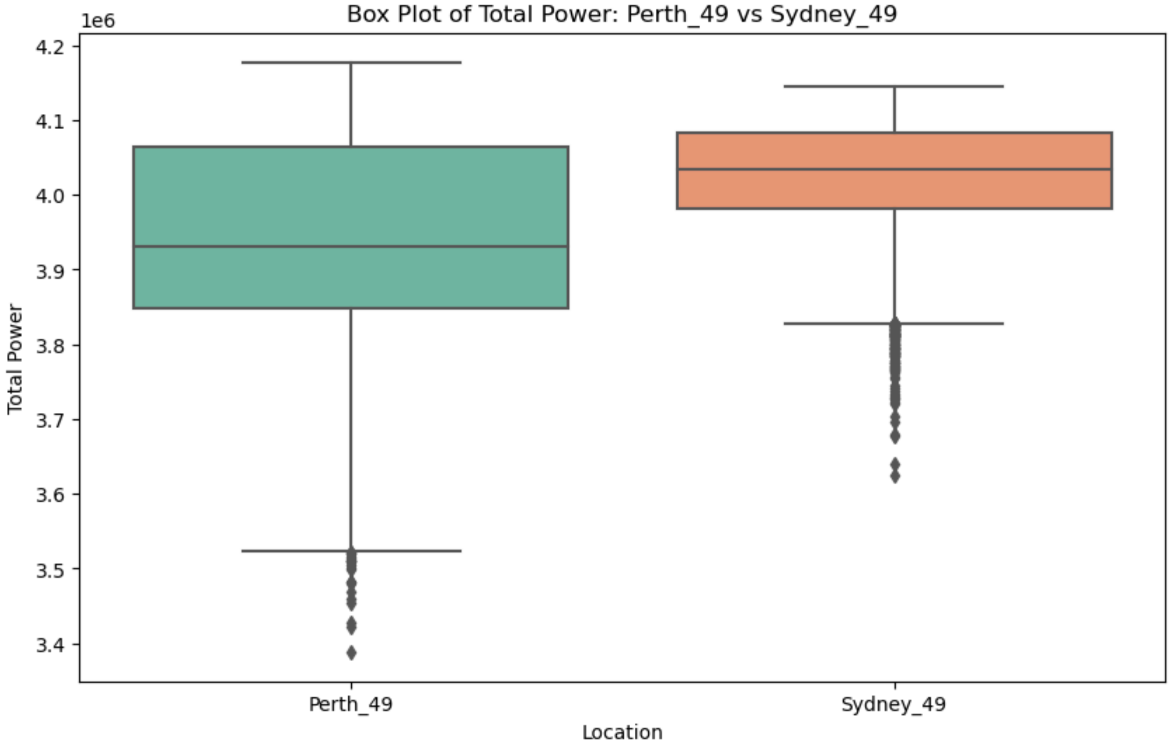
```
Training set size: (50881, 5), Testing set size: (12721, 5)
```

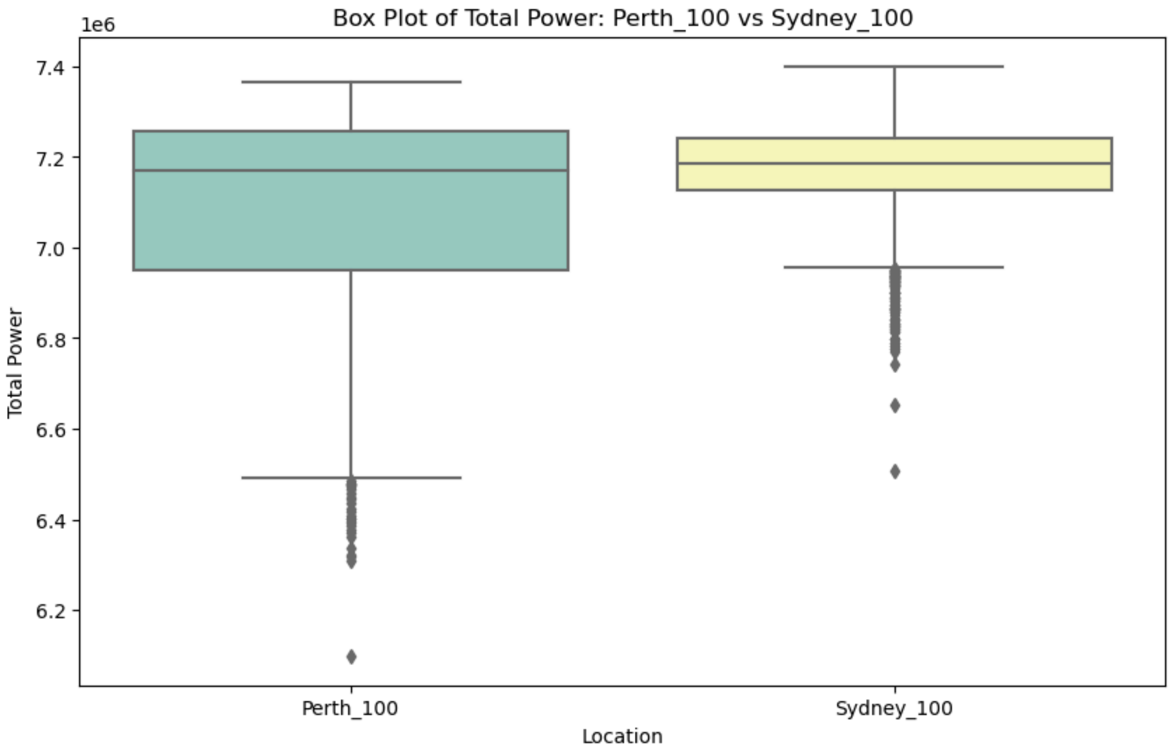
Data Visualisation :

#

In [26]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load or assume that datasets `perth_49`, `sydney_49`, `perth_100`, and `sydne

# Prepare the data for combined plotting
data_49 = pd.DataFrame({
    'Total_Power': pd.concat([perth_49['Total_Power'], sydney_49['Total_Power']
    'Location': ['Perth_49'] * len(perth_49) + ['Sydney_49'] * len(sydney_49)
})

data_100 = pd.DataFrame({
    'Total_Power': pd.concat([perth_100['Total_Power'], sydney_100['Total_Power
    'Location': ['Perth_100'] * len(perth_100) + ['Sydney_100'] * len(sydney_10
})

# Plot Perth_49 vs Sydney_49
plt.figure(figsize=(10, 6))
sns.boxplot(x='Location', y='Total_Power', data=data_49, palette="Set2")
plt.title('Box Plot of Total Power: Perth_49 vs Sydney_49')
plt.xlabel('Location')
plt.ylabel('Total Power')
plt.show()

# Plot Perth_100 vs Sydney_100
plt.figure(figsize=(10, 6))
sns.boxplot(x='Location', y='Total_Power', data=data_100, palette="Set3")
plt.title('Box Plot of Total Power: Perth_100 vs Sydney_100')
plt.xlabel('Location')
plt.ylabel('Total Power')
plt.show()
```

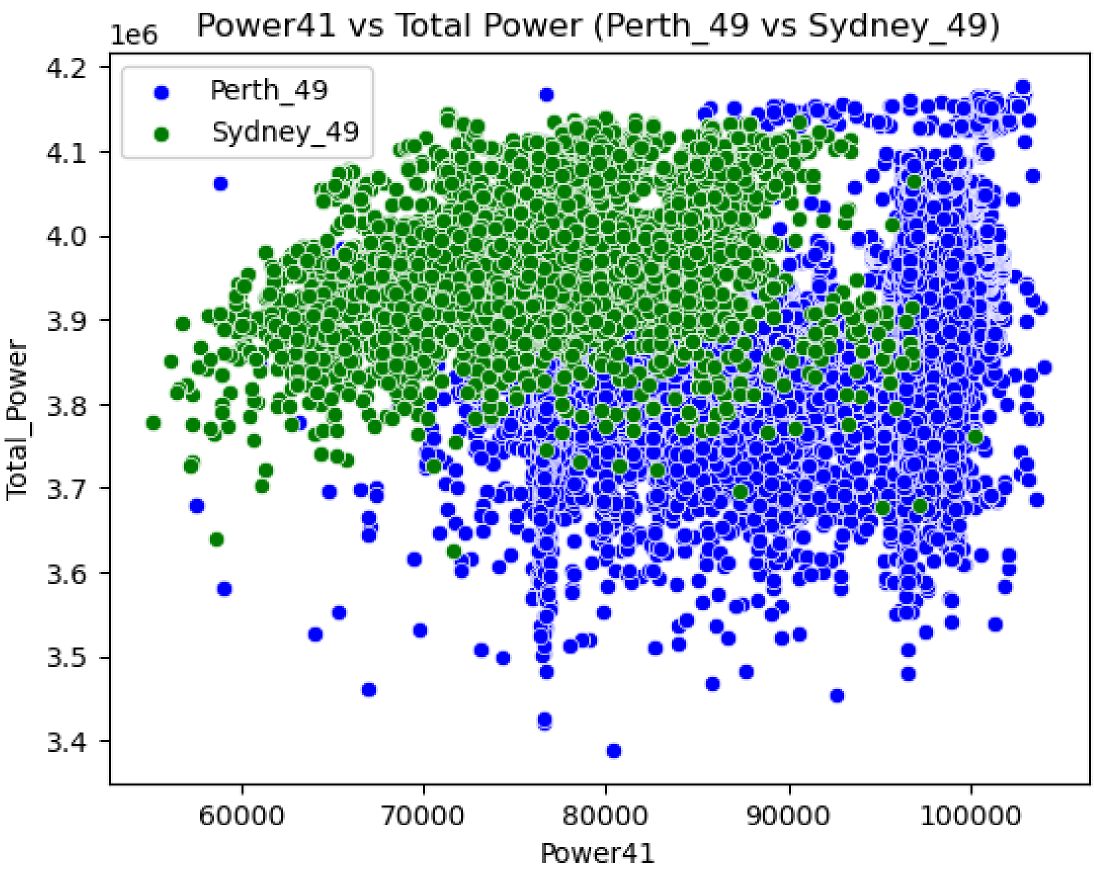


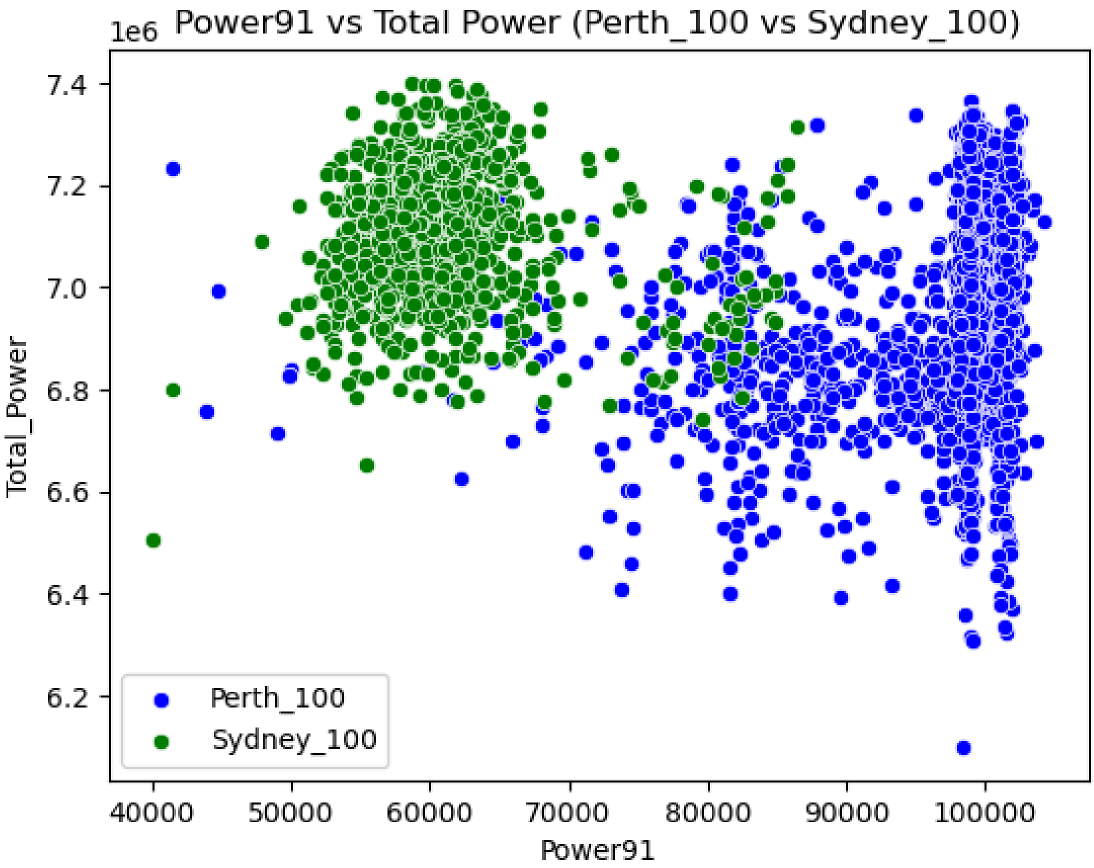


In [ ]:
```
Box Plot for Perth_49 vs Sydney_49 :

The box plot will display two vertical boxes, one for Perth_49 and one for Sydn

Box Plot for Perth_100 vs Sydney_100 :

This plot is similar to the first but compares the Total_Power values for Perth

Whether the median Total_Power is higher for Perth or Sydney in each dataset. T
```

Scatter Plot :

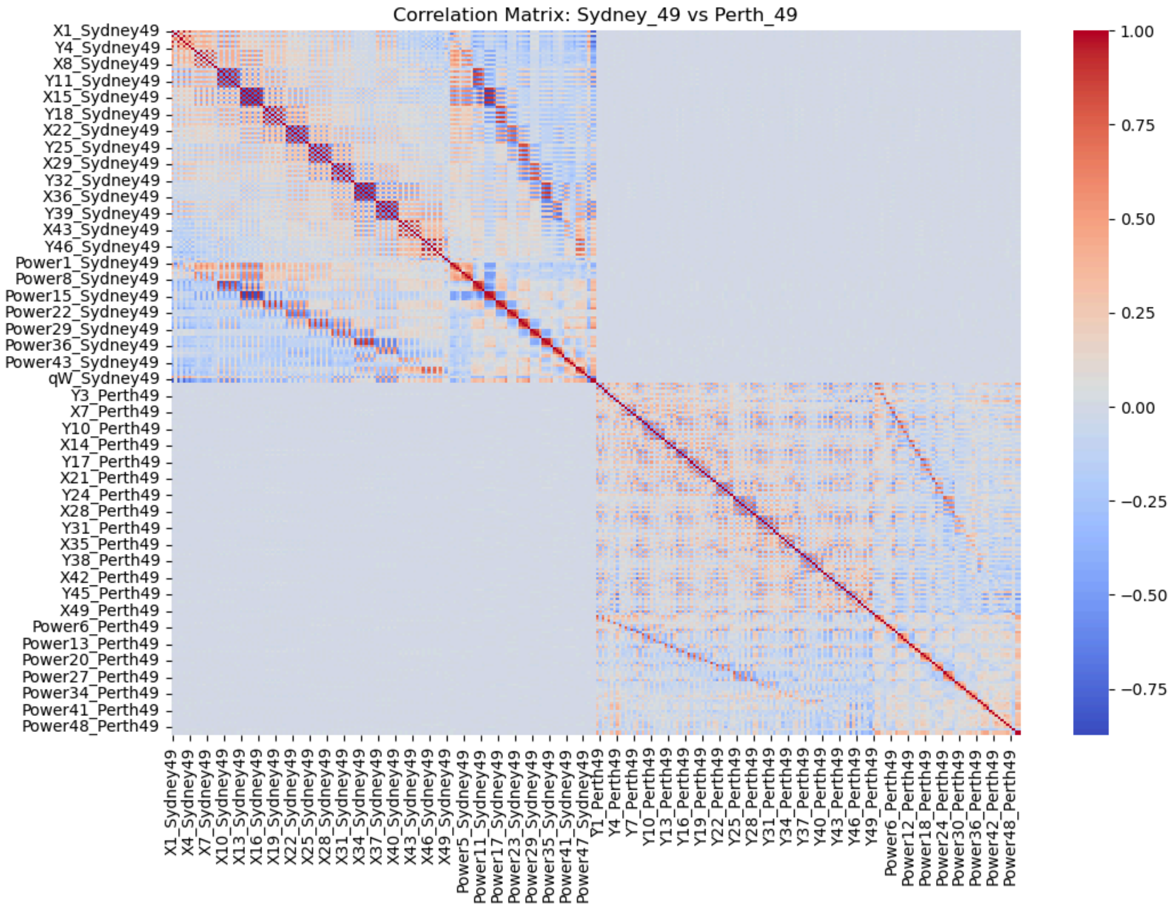
In [27]:
```
import seaborn as  sns
import matplotlib.pyplot as plt
# Scatter plot of Power41 vs Total_Power for Perth_49 and Sydney_49
sns.scatterplot(data=perth_49, x='Power41', y='Total_Power', color='blue', labe
sns.scatterplot(data=sydney_49, x='Power41', y='Total_Power', color='green', la
plt.title('Power41 vs Total Power (Perth_49 vs Sydney_49)')
plt.show()

#Scatter plot of Power91 vs Total_Power for Perth_100 and Sydney_100
sns.scatterplot(data=perth_100, x='Power91',y='Total_Power',color = 'blue',labe
sns.scatterplot(data=sydney_100, x='Power91',y='Total_Power',color ='green',lab
plt.title('Power91 vs Total Power (Perth_100 vs Sydney_100)')
plt.show()
```

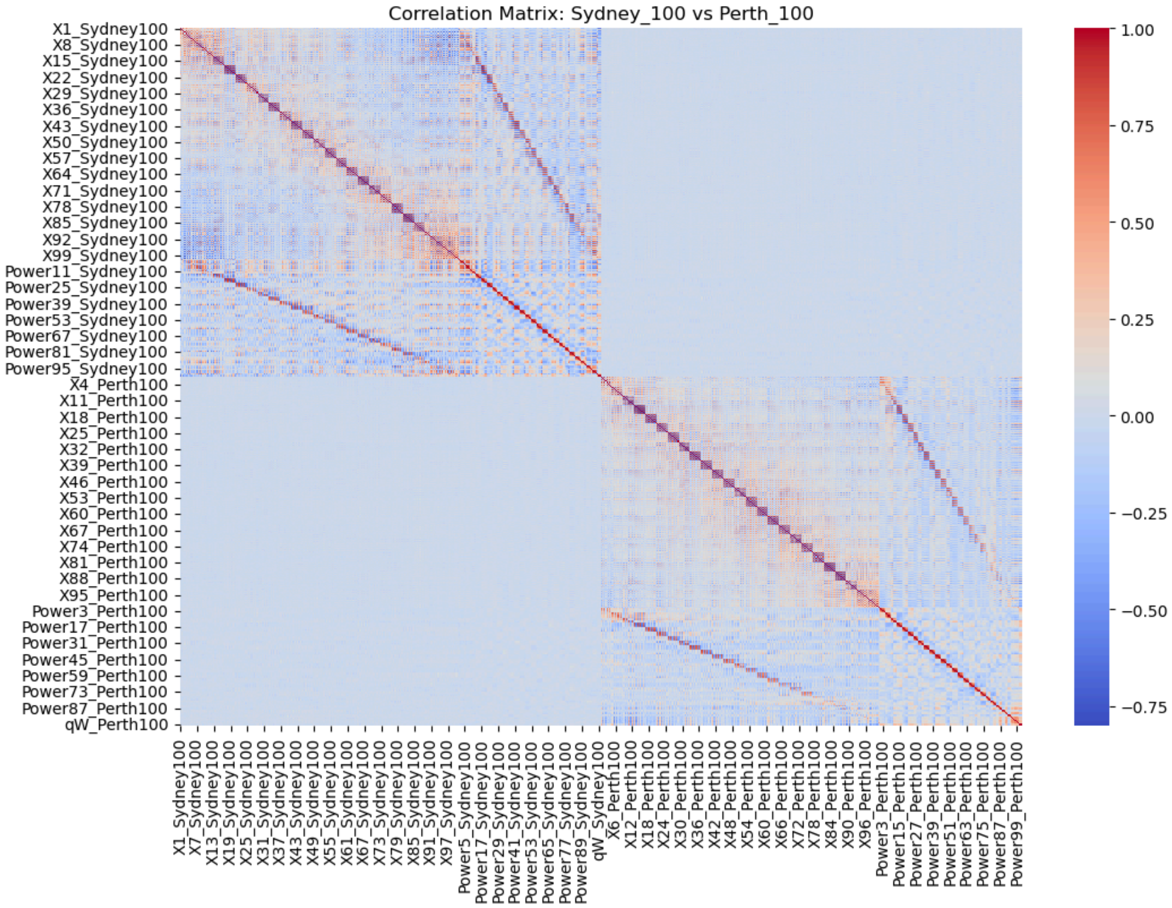




Correlation matrix :

In [28]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming sydney_49, perth_49, sydney_100, and perth_100 are loaded DataFrames
# Concatenate corresponding datasets and calculate correlations

# Sydney_49 vs Perth_49
combined_49 = pd.concat([sydney_49.add_suffix('_Sydney49'), perth_49.add_suffix
correlation_matrix_49 = combined_49.corr()

# Sydney_100 vs Perth_100
combined_100 = pd.concat([sydney_100.add_suffix('_Sydney100'), perth_100.add_su
correlation_matrix_100 = combined_100.corr()

# Plotting the correlation matrices
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix_49, annot=False, cmap='coolwarm', cbar=True)
plt.title('Correlation Matrix: Sydney_49 vs Perth_49')
plt.show()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix_100, annot=False, cmap='coolwarm', cbar=True)
plt.title('Correlation Matrix: Sydney_100 vs Perth_100')
plt.show()
```

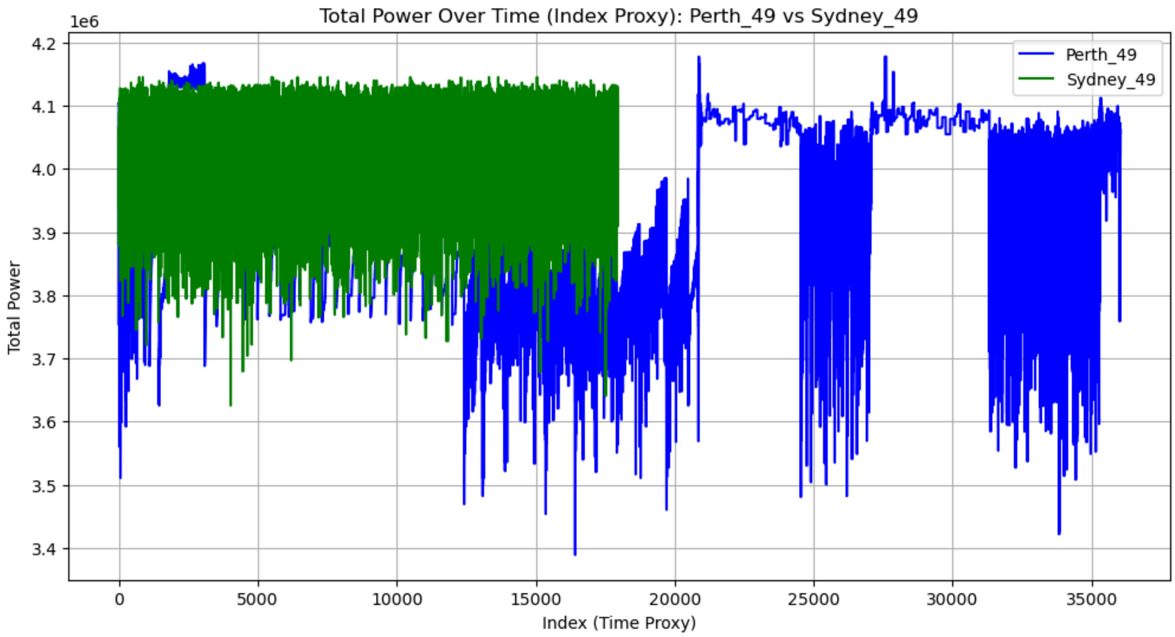


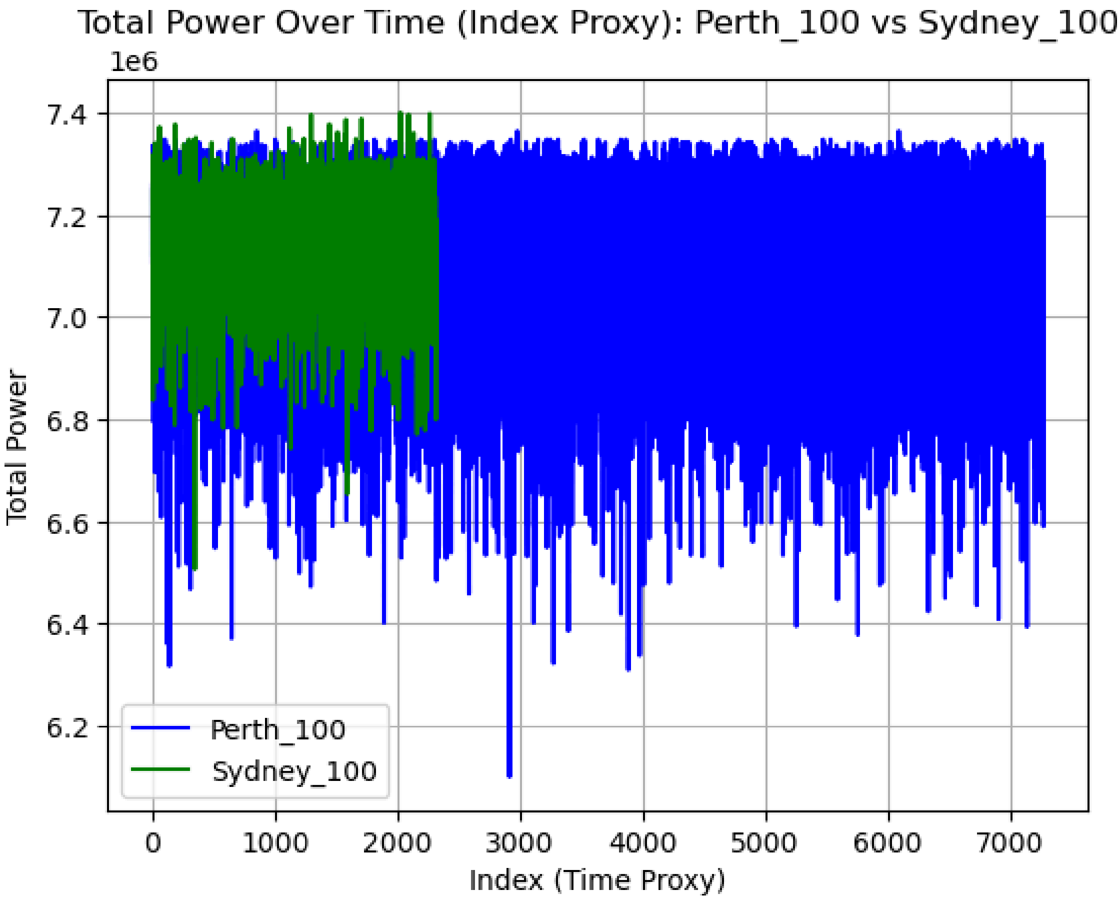


Line Plot :

In [29]:

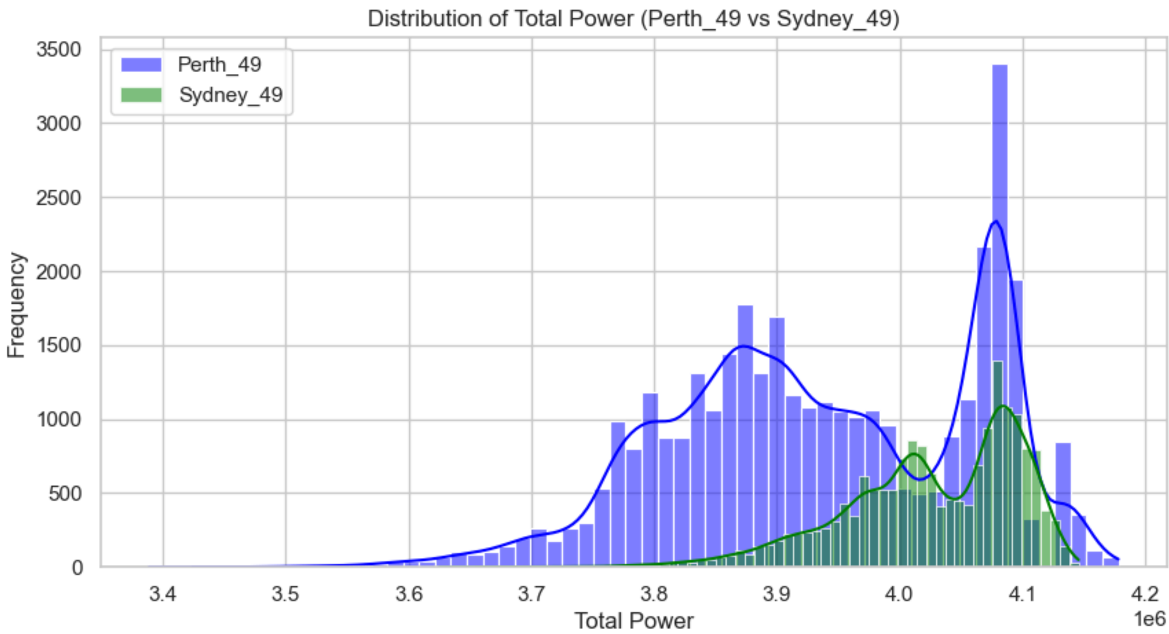
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming the row index represents time
plt.figure(figsize=(12, 6))

# Line plot for Perth_49
sns.lineplot(x=perth_49.index, y='Total_Power', data=perth_49, label='Perth_49'

# Line plot for Sydney_49
sns.lineplot(x=sydney_49.index, y='Total_Power', data=sydney_49, label='Sydney_

plt.title('Total Power Over Time (Index Proxy): Perth_49 vs Sydney_49')
plt.xlabel('Index (Time Proxy)')
plt.ylabel('Total Power')
plt.legend()
plt.grid(True)
plt.show()


# Line plot for Perth_49
sns.lineplot(x=perth_100.index, y='Total_Power', data=perth_100, label='Perth_1

# Line plot for Sydney_49
sns.lineplot(x=sydney_100.index, y='Total_Power', data=sydney_100, label='Sydne

plt.title('Total Power Over Time (Index Proxy): Perth_100 vs Sydney_100')
plt.xlabel('Index (Time Proxy)')
plt.ylabel('Total Power')
plt.legend()
plt.grid(True)
plt.show()
```

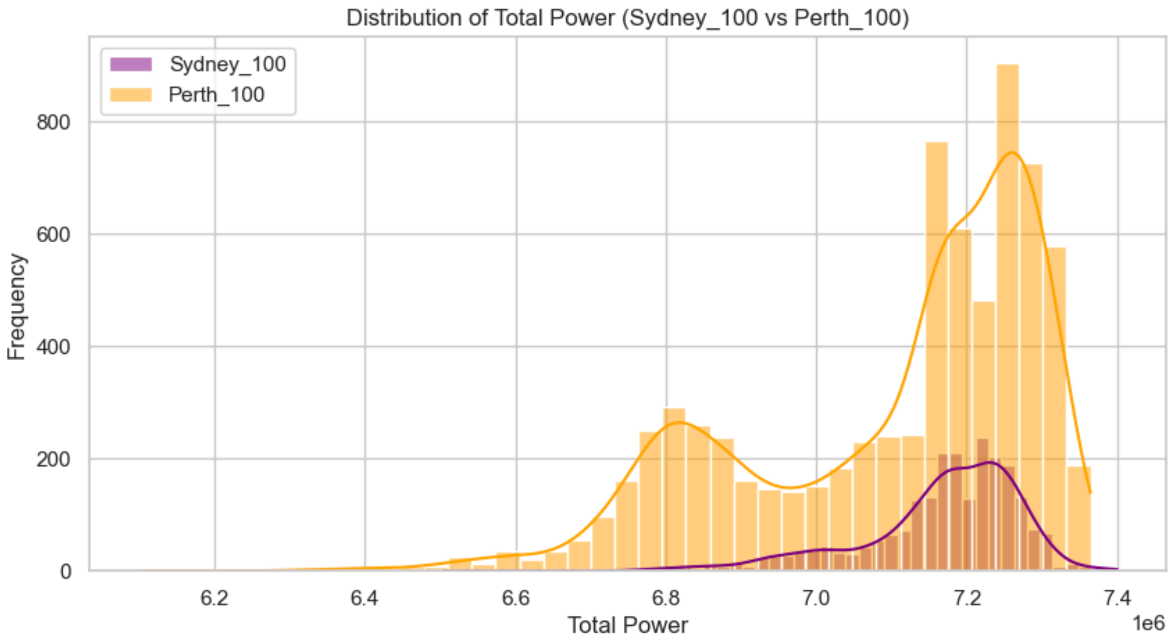



Total Power Over Time (Index Proxy): Perth_100 vs Sydney_100
1e6
7.4
7.2
7.0
6.8
6.6
6.4
6.2
Total Power
Perth_100
Sydney_100
0
1000
2000
3000
4000
5000
6000
7000
Index (Time Proxy)

Histogram :

In [30]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set the aesthetic style of the plots
sns.set(style='whitegrid')

# Histogram for Total Power in Perth_49 vs Sydney_49
plt.figure(figsize=(10, 5))
sns.histplot(perth_49['Total_Power'], kde=True, color='blue', label='Perth_49')
sns.histplot(sydney_49['Total_Power'], kde=True, color='green', label='Sydney_4
plt.legend()
plt.title('Distribution of Total Power (Perth_49 vs Sydney_49)')
plt.xlabel('Total Power')
plt.ylabel('Frequency')
plt.show()

# Histogram for Total Power in Sydney_100 vs Perth_100
plt.figure(figsize=(10, 5))
sns.histplot(sydney_100['Total_Power'], kde=True, color='purple', label='Sydney
sns.histplot(perth_100['Total_Power'], kde=True, color='orange', label='Perth_1
plt.legend()
plt.title('Distribution of Total Power (Sydney_100 vs Perth_100)')
plt.xlabel('Total Power')
plt.ylabel('Frequency')
plt.show()
```



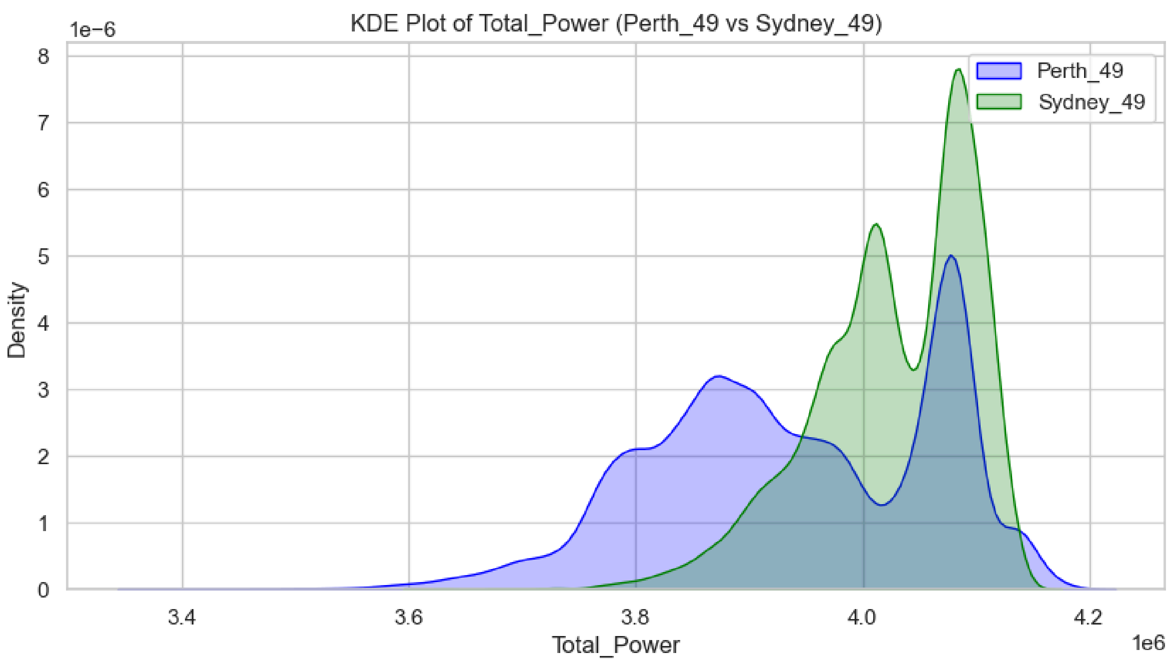

Distribution of Total Power (Sydney_100 vs Perth_100)
Sydney_100
Perth_100
800
600
400
200
0
Frequency
6.2
6.4
6.6
6.8
7.0
7.2
7.4
Total Power
1e6

In [ ]:

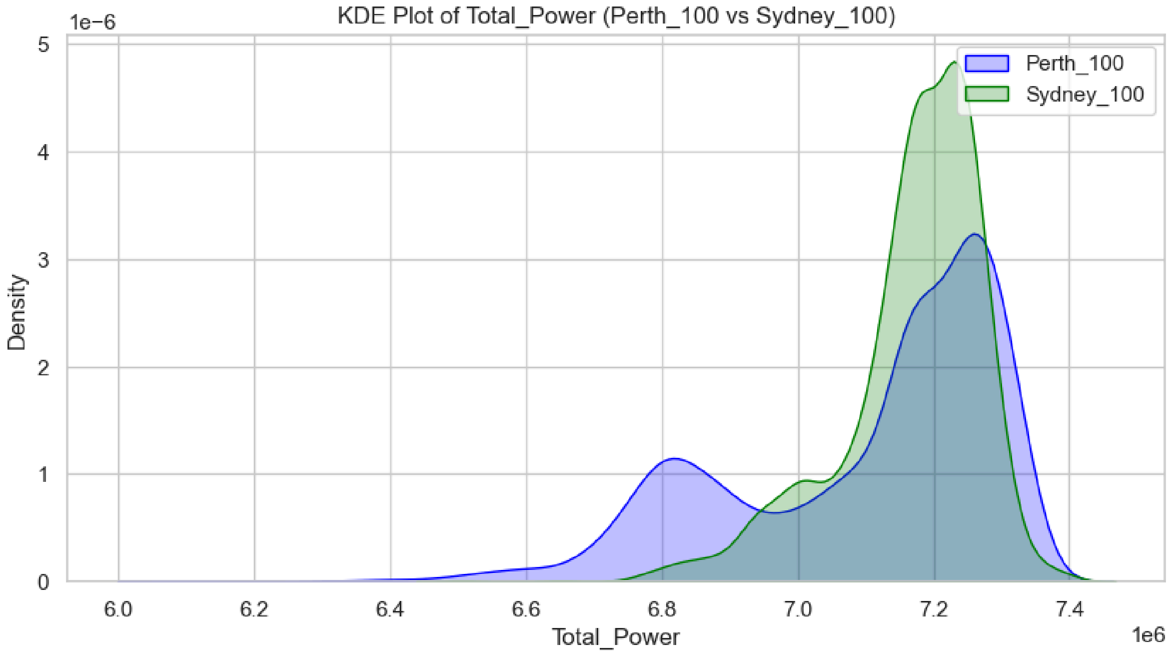
KDE Plot :

In [31]:
```
# KDE plot for Total_Power in both datasets
plt.figure(figsize=(10, 5))
sns.kdeplot(perth_49['Total_Power'], shade=True, color='blue', label='Perth_49'
sns.kdeplot(sydney_49['Total_Power'], shade=True, color='green', label='Sydney_
plt.legend()
plt.title('KDE Plot of Total_Power (Perth_49 vs Sydney_49)')
plt.show()

plt.figure(figsize=(10, 5))
sns.kdeplot(perth_100['Total_Power'], shade=True, color='blue', label='Perth_10
sns.kdeplot(sydney_100['Total_Power'], shade=True, color='green', label='Sydney
plt.legend()
plt.title('KDE Plot of Total_Power (Perth_100 vs Sydney_100)')
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_6752\4130300428.py:3: FutureWarnin
g:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(perth_49['Total_Power'], shade=True, color='blue', label='Perth_
49')
C:\Users\Admin\AppData\Local\Temp\ipykernel_6752\4130300428.py:4: FutureWarnin
g:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(sydney_49['Total_Power'], shade=True, color='green', label='Sydn
ey_49')
```



```
C:\Users\Admin\AppData\Local\Temp\ipykernel_6752\4130300428.py:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(perth_100['Total_Power'], shade=True, color='blue', label='Perth_100')
C:\Users\Admin\AppData\Local\Temp\ipykernel_6752\4130300428.py:11: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(sydney_100['Total_Power'], shade=True, color='green', label='Sydney_100')
```



In [ ]:
```
#  PIPELINING THE MODEL:
```

The pipeline, including preprocessing steps and trained models, will be saved as 'preprocessing_model_pipeline.pkl'.

In [ ]:
```
Model evaluation metrics and predictions :
Evaluation Metrics:
Mean Absolute Error (MAE): Measures the average magnitude of errors in the pred
```

In [ ]:
```
HYPER PARAMETER TUNING :
Hyperparameter tuning is essential in this project because it helps optimize th
```

Save the Model :

The pipeline model is saved to a pickle file.

In [ ]:

```
Basic Insights: Perth vs Sydney Power Consumption Data
Here are some key takeaways from the analysis of power consumption in Perth and

1.Sydney's Power Consumption is Higher:
Across all datasets, Sydney consistently shows higher average power consumption

2.Variability in Power Consumption:
Sydney displays more variability in power usage, with a wider range of values c

3.Outliers :
were detected using the Z-score method. These outliers represent abnormal power

4.Distribution of Power Consumption:
The distribution of power usage in Perth is more tightly packed around specific

5.Key Features Driving Power Usage:
Random Forest feature importance analysis revealed that certain time intervals

6.Comparison Between Perth_49, Sydney_49, Perth_100, and Sydney_100:
The box plot comparison shows that the Sydney_100 dataset had the widest range

These insights provide a foundation for understanding the power consumption beh
```

In [ ]: