

# **MEASURE ENERGY CONSUMPTION**

**NAME:** LINGESH.G

**NM ID:** aut22211302

**Phase 3 document**

**Development Part - 1**



## **INTRODUCTION :**

- The development of measuring energy consumption is a critical and rapidly evolving field that plays a pivotal role in our efforts to manage resources efficiently, reduce carbon emissions, and promote sustainability.
- Measuring energy consumption has historically been a fundamental practice for both individuals and industries. It provides essential insights into resource allocation, billing, and environmental impact.
- Smart meters, advanced sensors, and data analytics have revolutionized our ability to track and manage energy use in real time, making it a cornerstone of the modern energy landscape.
- This introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the energy dataset, and perform critical preprocessing steps.
- Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis.

## GIVEN DATASET:-

A good data source for hourly energy consumption using machine should be accurate and complete.

	count	mean	std	min	25%	50%	7
AEP	121273.0	15499.513717	2591.399065	9581.0	13630.0	15310.0	1
COMED	66497.0	11420.152112	2304.139517	7237.0	9780.0	11152.0	1
DAYTON	121275.0	2037.851140	393.403153	982.0	1749.0	2009.0	2
DEOK	57739.0	3105.096486	599.859026	907.0	2687.0	3013.0	3
DOM	116189.0	10949.203625	2413.946569	1253.0	9322.0	10501.0	1
DUQ	119068.0	1658.820296	301.740640	1014.0	1444.0	1630.0	1
EKPC	45334.0	1464.218423	378.868404	514.0	1185.0	1386.0	1
FE	62874.0	7792.159064	1331.268006	0.0	6807.0	7700.0	8
NI	58450.0	11701.682943	2371.498701	7003.0	9954.0	11521.0	1
PJME	145366.0	32080.222831	6464.012166	14544.0	27573.0	31421.0	3
PJMW	143206.0	5602.375089	979.142872	487.0	4907.0	5530.0	6
PJM_Load	32896.0	29766.427408	5849.769954	17461.0	25473.0	29655.0	3

## Necessary steps to follow:

### 1.Import Libraries:

For analysing,Preprocessing and visualizing the dataset we using some python libraries.

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
import seaborn as sns
```

```
import xgboost as xgb
```

```
from sklearn.metrics import mean_squared_error
```

```
color_pal = sns.color_palette()
```

```
plt.style.use('fivethirtyeight')
```

## 2.importing the dataset:

```
energy_df = pd.read_csv("C:\\Users\\VIJAY\\Desktop\\PJMw_hourly.csv")
```

```
energy_df.head()
```

## 3.Data cleaning:

Getting information about the dataset

```
energy_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 143206 entries, 12/31/2002 01:00 to 1/2/2018 00:00
Data columns (total 1 columns):
#   Column   Non-Null Count  Dtype
---  -
0   PJMW_MW  143206 non-null  int64
dtypes: int64(1)
memory usage: 2.2+ MB
```

## 4.Exploratory Data Analysis:

```
energy_df = energy_df.set_index('Datetime')
```

```
energy_df.head()
```

## 5. Check for missing values:

```
energy_df.isnull().sum()
```

```
PJMW_MW    0
dtype: int64
```

## 6. Shape of Dataset:

```
energy_df.shape
```

```
(143206, 1)
```

## 7. Data Splitting:

```
train = energy_df.loc[energy_df.index < '01-01-2015']  
test = energy_df.loc[energy_df.index >= '01-01-2015']  
print(train.shape, test.shape)
```

```
(111766, 1) (31440, 1)
```

## 8. Feature Creation:

```
energy_df = create_features(energy_df)  
energy_df.head()
```

	PJMW_MW	hour	dayofweek	quarter	month	year	dayofyear
Datetime							
2002-12-31 01:00:00	5077.0	1	1	4	12	2002	365
2002-12-31 02:00:00	4939.0	2	1	4	12	2002	365
2002-12-31 03:00:00	4885.0	3	1	4	12	2002	365
2002-12-31 04:00:00	4857.0	4	1	4	12	2002	365
2002-12-31 05:00:00	4930.0	5	1	4	12	2002	365

## Importance of loading and processing dataset:

- Loading and preprocessing the dataset is an important first step in building any machine learning model.
- However, it is especially important for energy consumption models, as measuring consumption of energy datasets are often complex.

- By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

### Challenges involved in loading and preprocessing dataset:-

- Handling missing values: Common methods for handling missing values include dropping the rows with missing values, imputing the missing values with the mean or median of the feature, or using a more sophisticated method such as multiple imputation.
- Encoding categorical variables: Energy consumption datasets often contain categorical features. These features need to be encoded before they can be used by machine learning models.
- Splitting the dataset into training and testing sets: Once the data has been pre-processed, we need to split the dataset into training and testing sets. It is important to split the dataset in a way that is representative of the real world distribution of the data.

### How to overcome these challenges:-

- Carefully consider the specific needs of your model: The best way to preprocess the data will depend on the specific machine learning algorithm that you are using. It is important to carefully consider the

requirements of the algorithm and to preprocess the data in a way that is compatible with the algorithm.

- Validate the preprocessed data: It is important to validate the preprocessed data to ensure that it is in a format that can be used by the machine learning algorithm and that it is of high quality. This can be done by inspecting the data visually or by using statistical methods.

## **1. Loading the dataset:**

- ✓ Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
- ✓ The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used.
- ✓ However, there are some general steps that are common to most machine learning frameworks:

a) Identify the dataset: The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

b) Load the dataset: Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

c) Preprocess the dataset: Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.

### Program:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


RED="\033[91m"

GREEN="\033[92m"

YELLOW="\033[93m"

BLUE="\033[94m"

RESET="\033[0m"


#LOAD DATASET

df=pd.read_csv("C:\\Users\\VIJAY\\Desktop\\PJMw_hourly.csv")

df["Datetime"]=pd.to_datetime(df["Datetime"])


#DATA CLEANING

print(BLUE +"\n DATA CLEANING" +RESET)
```



```
#Check for missing values
```

```
missing_values=df.isnull().sum()
```

```
print(GREEN + "Missing values:" +RESET)
```

```
print(missing_values)
```

## **#HANDLE MISSING VALUES**

```
df.dropna(inplace=True)
```

## **#CHECK FOR DUPLICATE VALUES**

```
duplicate_values=df.duplicated().sum()
```

```
print(GREEN + "Duplicate Values :"+RESET)
```

```
print(duplicate_values)
```

## **#DROP DUPLICATE VALUES**

```
df.drop_duplicates(inplace=True)
```

## **#DATA ANALYSIS**

```
print(BLUE + "\n DATA ANALYSIS"+RESET)
```

## **#SUMMARY STATISTICS**

```
summary_stats=df.describe()
```

```
print(GREEN + "Summary statistics :"+RESET)
```

```
print(summary_stats)
```

## **#DATA VISUALIZATION**

```
#line plot for energy consumption over time
```

```
plt.figure(figsize=(12,6))
```

```
plt.plot(df.index,df["PJM_W_MW"],label="Energy consumption(PJM_W_MW)")
```

```
plt.xlabel("Datetime")
```

```
plt.ylabel("Energy consumption(MW)")
```

```
plt.title("Energy consumption Over Time")
```

```
plt.grid()
```

```
plt.legend()
```

```
plt.show()
```

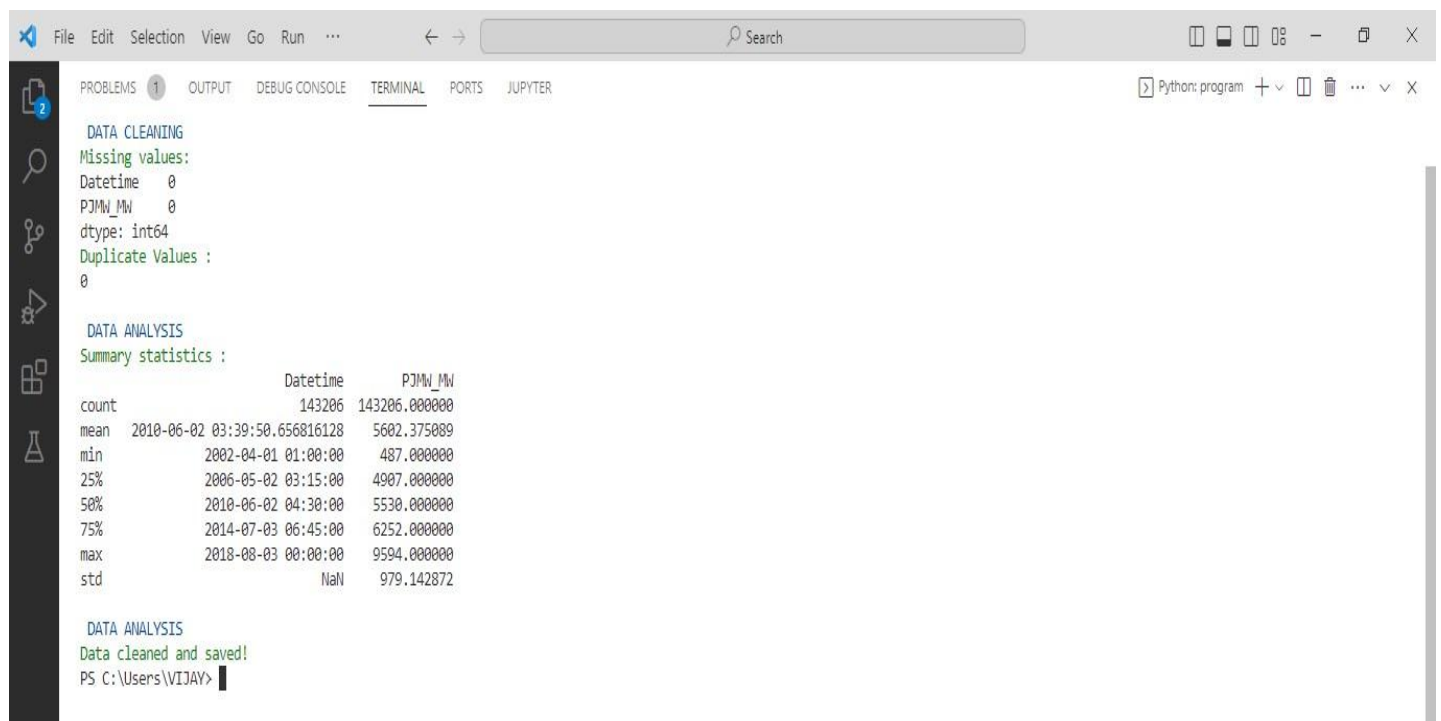
## #SAVING THE FILE

```
df.to_csv("cleaned_PJM_W_hourly.csv",index=False)
```

```
print(BLUE+"\n DATA ANALYSIS" +RESET)
```

```
PRINT(GREEN +"Data cleaned and saved!" +RESET)
```

## Output:



The screenshot shows a Jupyter Notebook interface with a terminal window open. The terminal displays the following output:

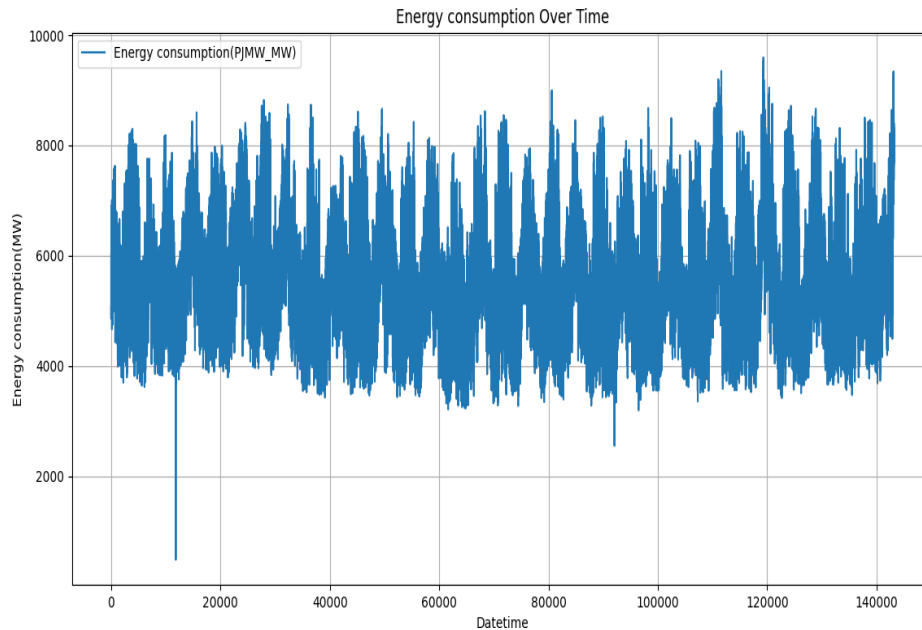
```
DATA CLEANING
Missing values:
Datetime      0
PJM_W_MW      0
dtype: int64
Duplicate Values :
0

DATA ANALYSIS
Summary statistics :

```

	Datetime	PJM_W_MW
count	143206	143206.000000
mean	2010-06-02 03:39:50.656816128	5602.375009
min	2002-04-01 01:00:00	487.000000
25%	2006-05-02 03:15:00	4907.000000
50%	2010-06-02 04:30:00	5530.000000
75%	2014-07-03 06:45:00	6252.000000
max	2018-08-03 00:00:00	9594.000000
std	NaN	979.142872

```
DATA ANALYSIS
Data cleaned and saved!
PS C:\Users\VIJAY>
```



## **2. Preprocessing the dataset:-**

- Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.
- This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

## **Visualization and Pre-Processing of Data:**

Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.

## 1. Visualization:-

- Data visualization is a powerful tool for understanding and analyzing energy consumption patterns. Visualizations can help identify trends, anomalies, and areas for improvement in energy consumption.
- Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs and maps.
- Some python libraries for data visualization
  - Matplotlib
  - Seaborn
  - numpy
  - XGboost
  - Pandas

### **#data visualization**

#### **#line plot for energy consumption over time**

```
plt.figure(figsize=(12,6))  
  
plt.plot(df.index,df["PJM_W_MW"],label="Energy consumption(PJM_W_MW)")  
  
plt.xlabel("Datetime")  
  
plt.ylabel("Energy consumption(MW)")  
  
plt.title("Energy consumption Over Time")  
  
plt.grid()
```

```
plt.legend()
```

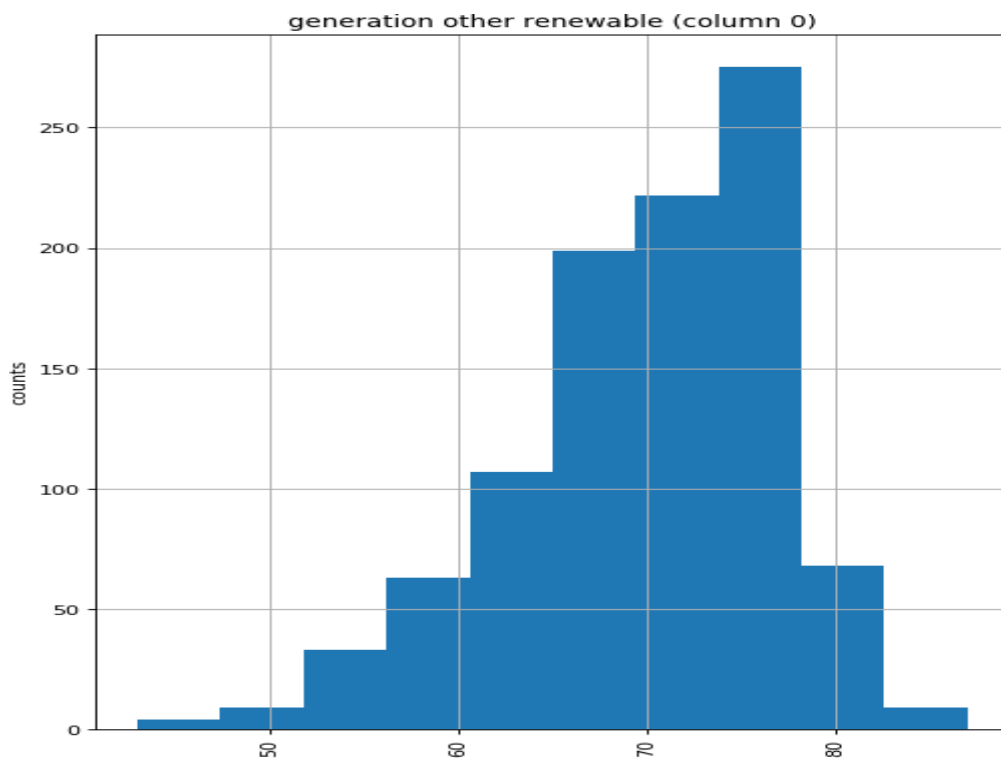
```
plt.show()
```

### **Program:**

In [1]:

```
plotPerColumnDistribution(df1, 10, 5)
```

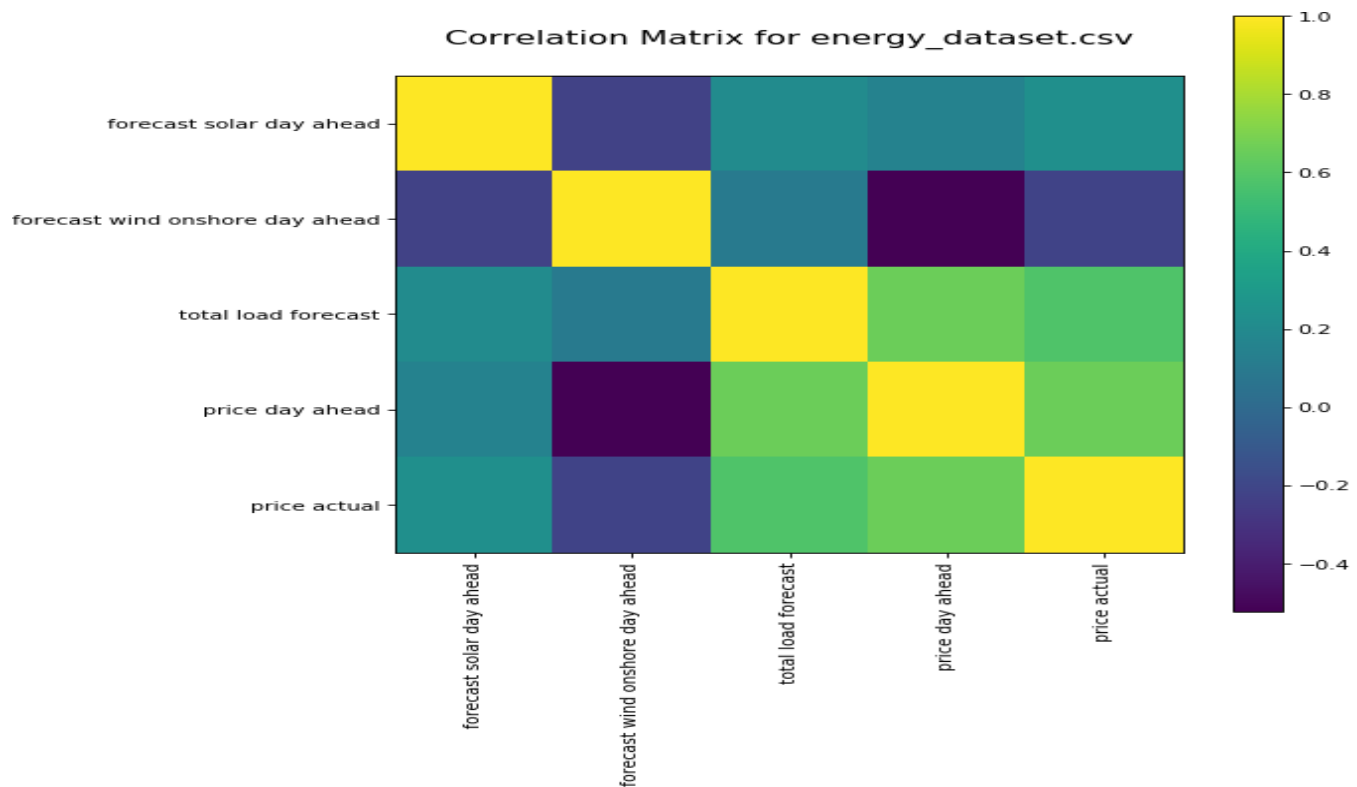
Out [1]:



In [2]:

```
PlotCorrelationMatrix(df1, 8)
```

Out [2]:



In [3]:

```
plotScatterMatrix(df1, 20, 10)
```

#Scatter plots are useful for visualizing the relationship between two continuous variables, such as energy consumption and time.

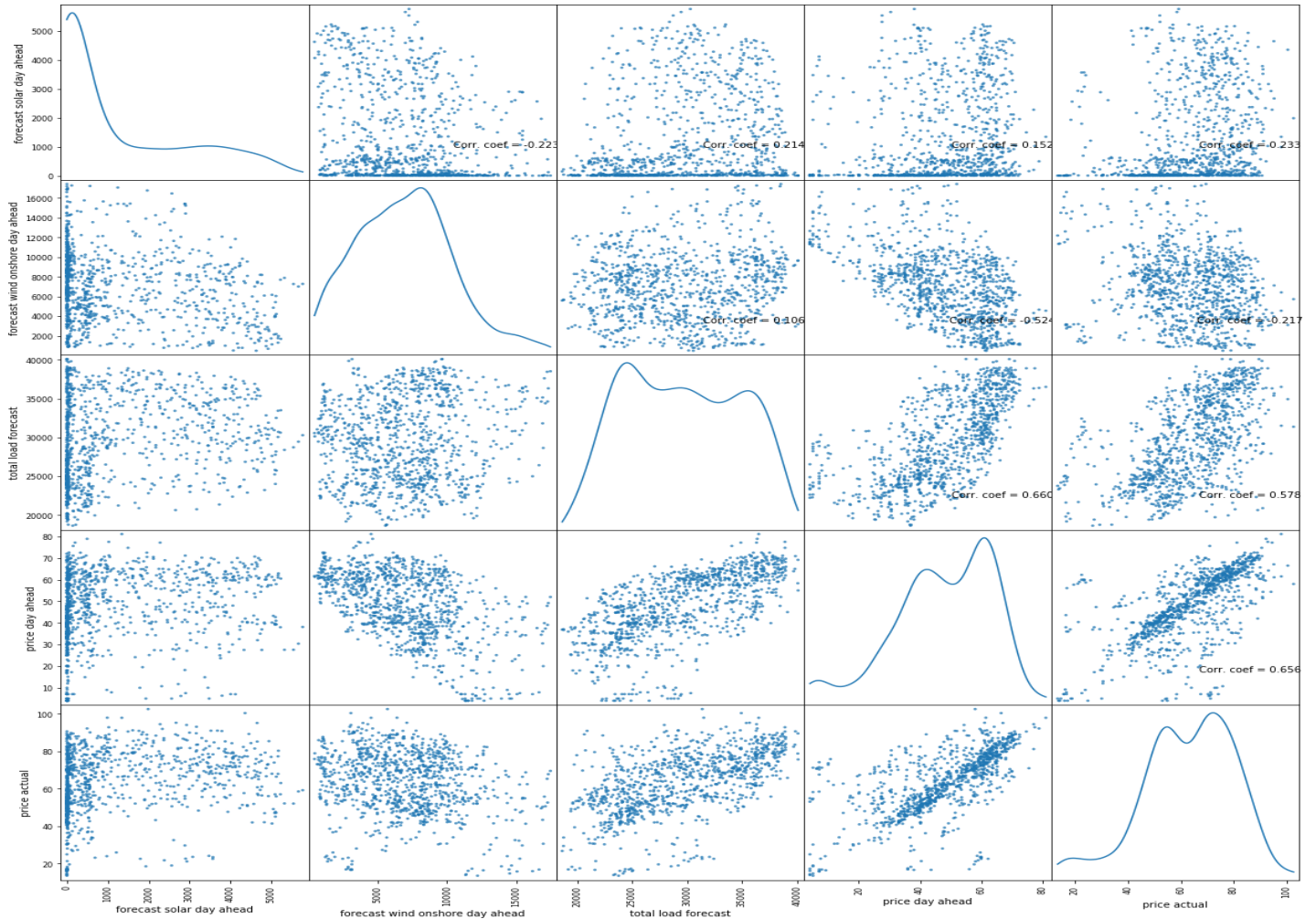
**X-axis:** Time (e.g., days, months, or years).

**Y-axis:** Energy consumption.

# Density plots, also known as kernel density plots, provide a smoothed representation of the data's distribution.

Out [3]:

Scatter and Density Plot

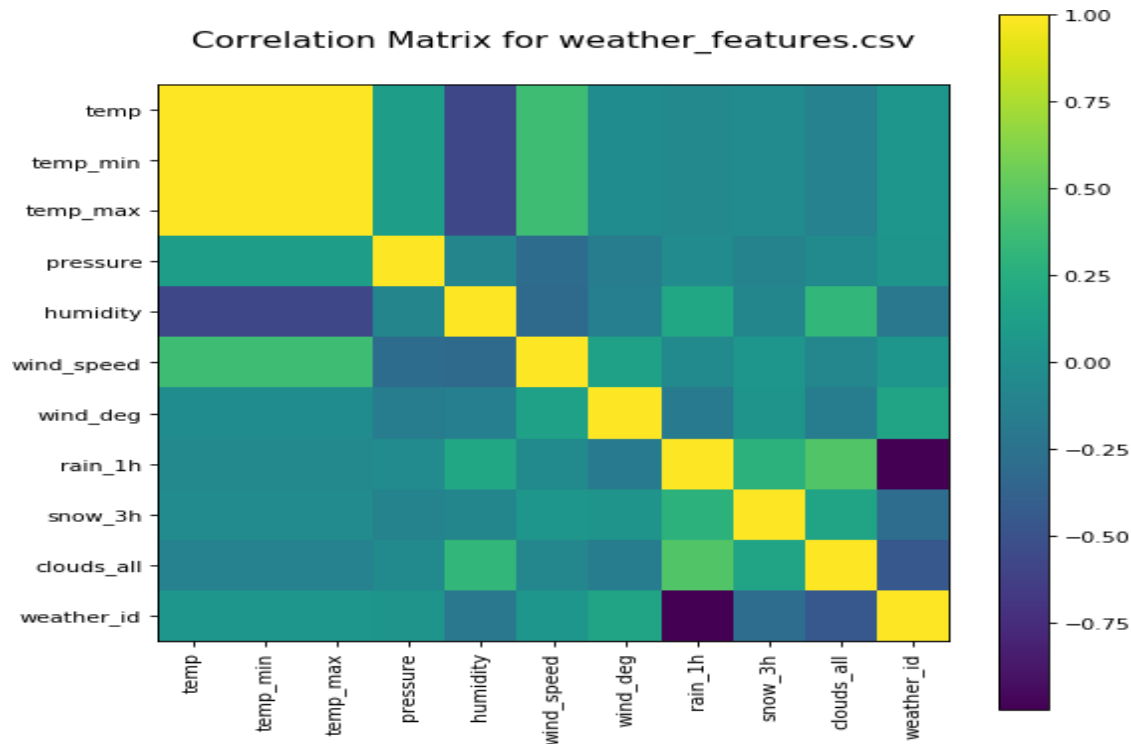


In [4]:

```
plotCorrelationMatrix(df2, 8)
```

# A below diagram represents correlation matrix for weather features.

Out [4]:



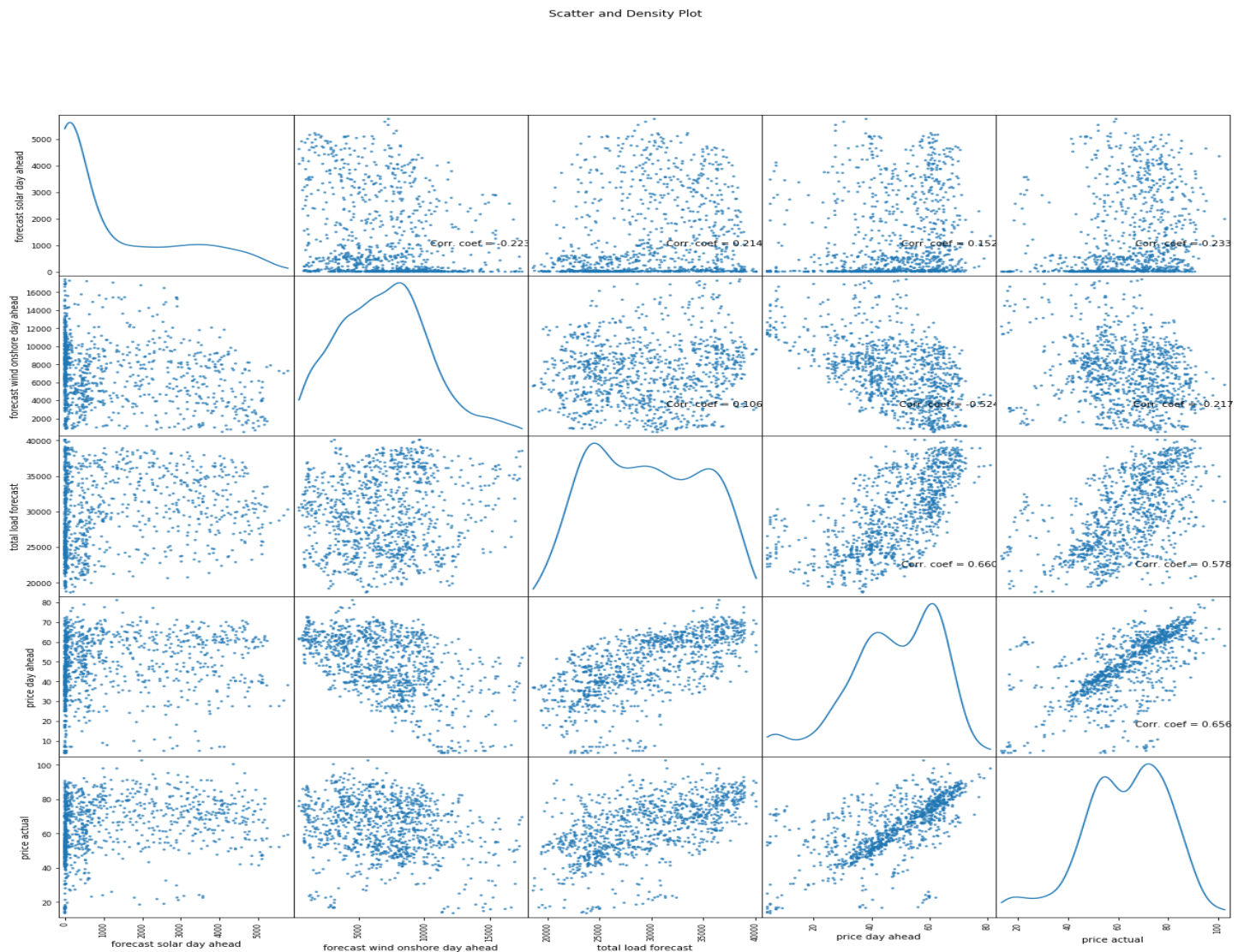
In [5]:

```
plotScatterMatrix(df1, 20, 10)
```

1. We import the necessary libraries: Matplotlib for creating plots and Seaborn for the density plot.
2. We define sample energy consumption data. You should replace this with your actual data.
3. We create a scatter plot using `plt.scatter()`. This plots the energy consumption data points against their index.



Out [5]:

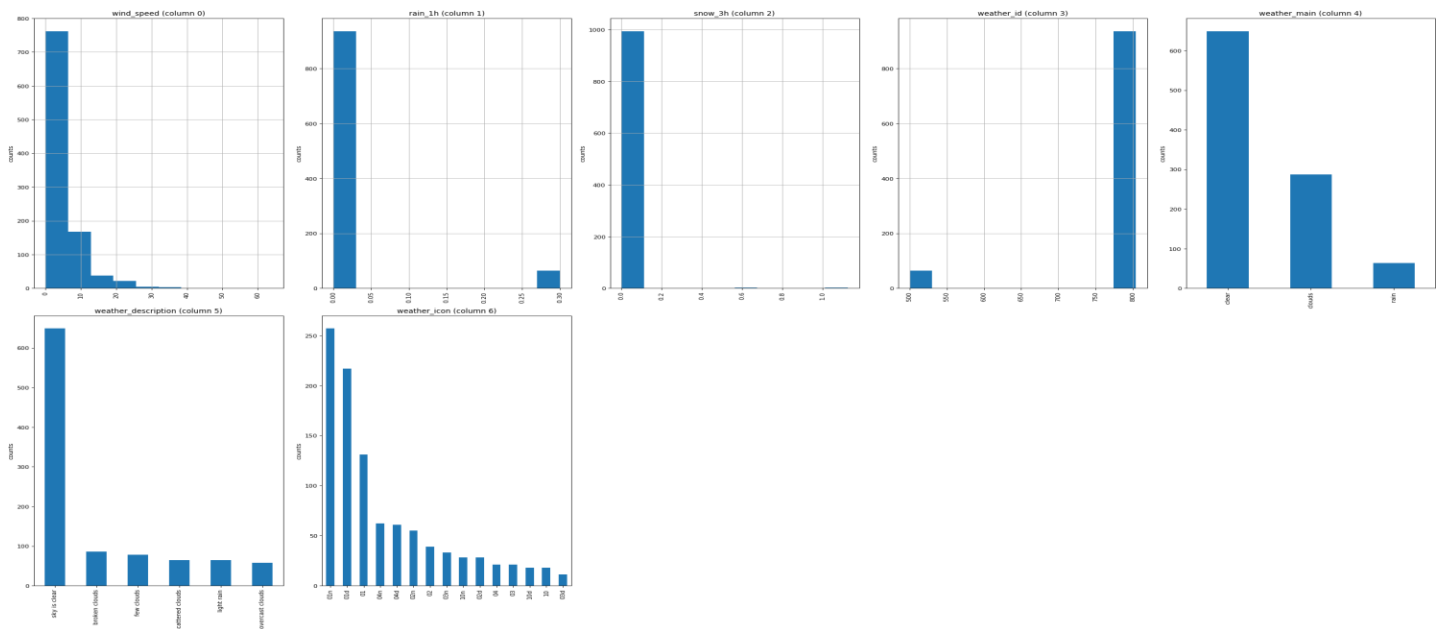


In [6]:

```
plotPerColumnDistribution(df2, 10, 5)
```

# To create distribution plots for each column in your energy consumption dataset, you can use Seaborn to generate histograms or kernel density plots for each column separately.

Out [6]:



In [7]:

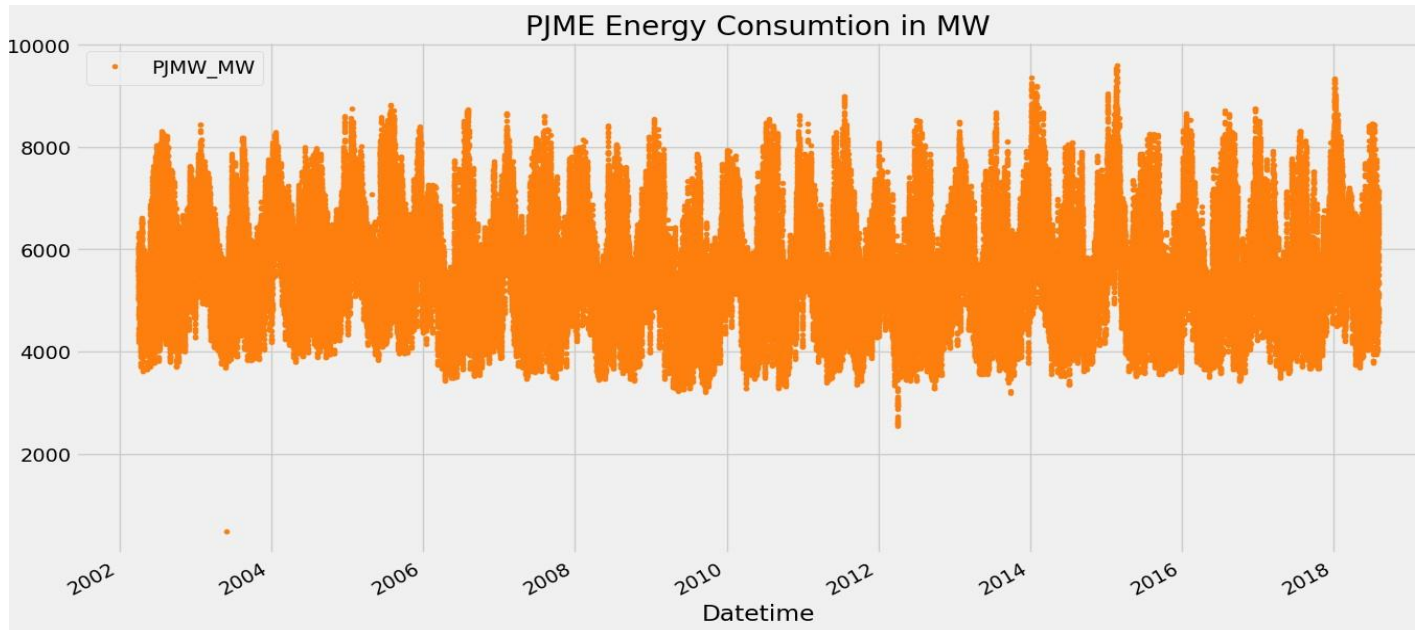
```
energy_df.index = pd.to_datetime(energy_df.index)
```

```
energy_df.plot(style='.', figsize=(15, 8), color=color_pal[1],  
               title='PJME Energy Consumption in MW');
```

#points

1. Analyzing the energy consumption data for PJME (PJM Interconnection), which is an electric grid operator in the United States.
2. To work with PJME energy consumption data, you'll need to have a dataset containing relevant information.

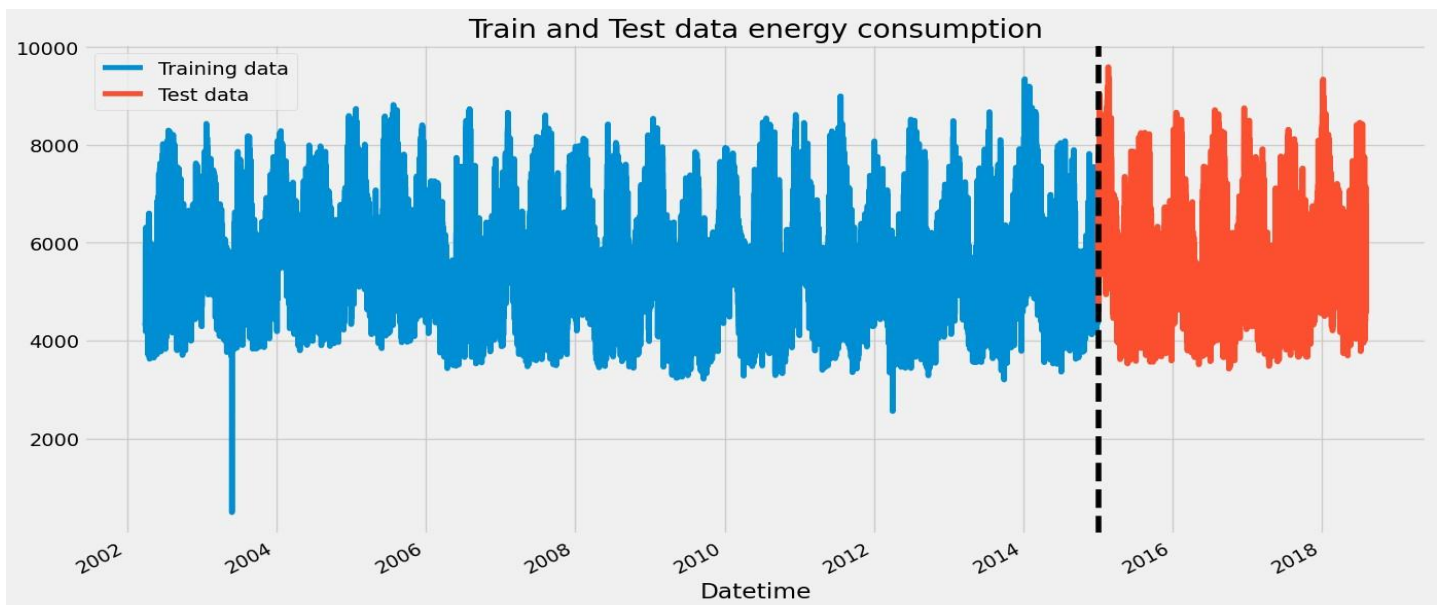
Out [7]:



In [8]:

```
fig, ax = plt.subplots(figsize=(15,8))
train.plot(ax=ax, label='Training data')
test.plot(ax=ax, label='Test data')
ax.axvline('01-01-2015', color='black', ls='--')
ax.legend(['Training data', 'Test data'])
plt.title('Train and Test data energy consumption')
plt.show()
```

Out [8]:



In [9]:

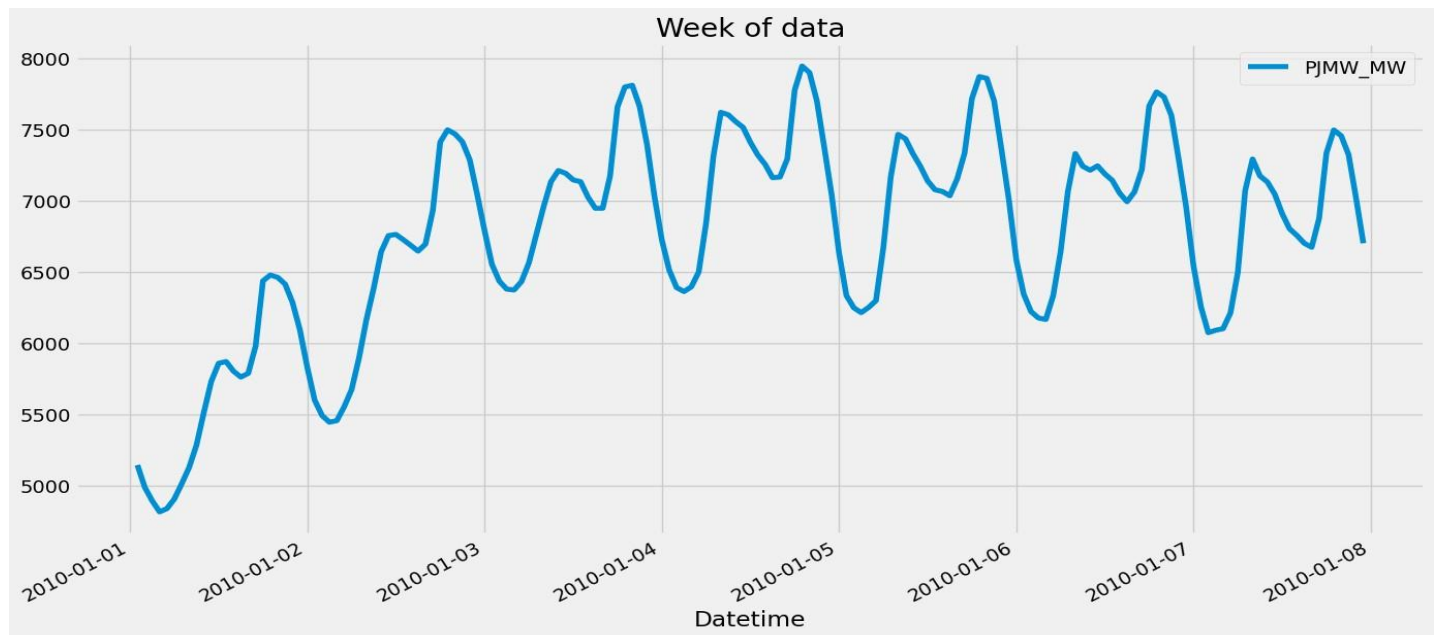
```
energy_df.loc[(energy_df.index > '01-01-2010') & (energy_df.index < '01-08-2010')].plot(figsize=(15,8), title='Week of data')
```

```
plt.show()
```

#The below diagram represents the week of data of energy consumption in home or buildings.

In PJMW\_MW dataset it works based Datetime.

Out [9]:



In [10]:

```
energy_df = create_features(energy_df)
```

```
energy_df.
```

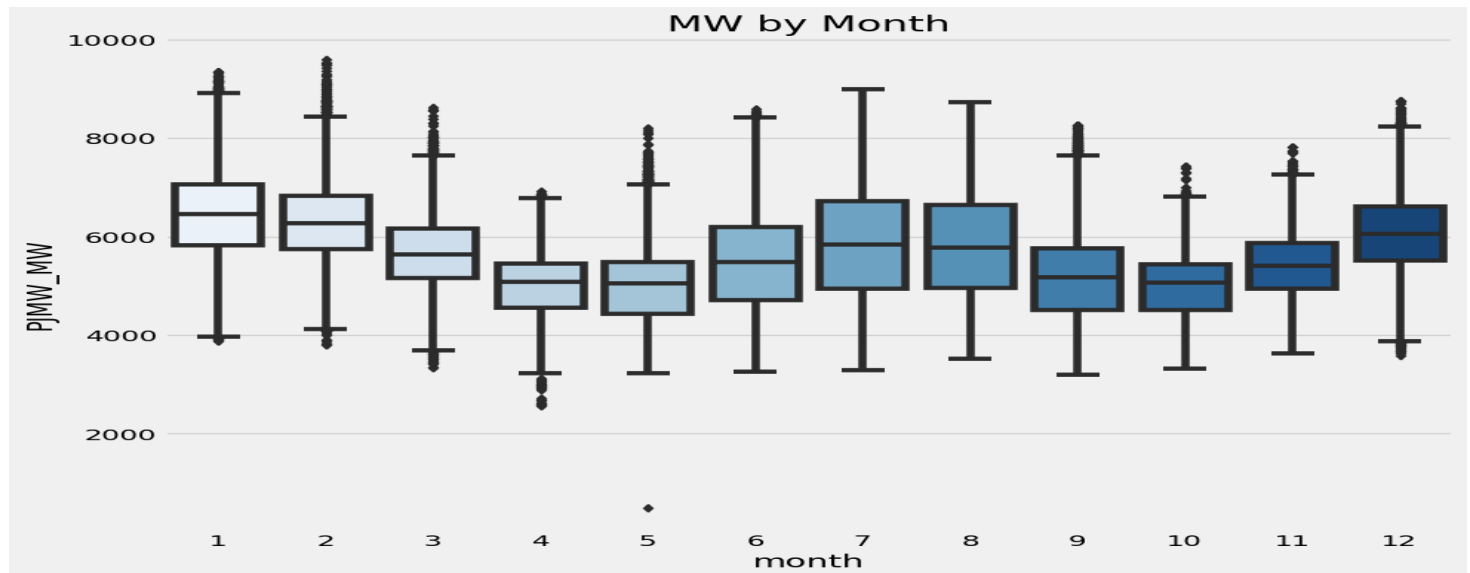
	PJM_W_MW	hour	dayofweek	quarter	month	year	dayofyear
Datetime							
2002-12-31 01:00:00	5077.0	1	1	4	12	2002	365
2002-12-31 02:00:00	4939.0	2	1	4	12	2002	365
2002-12-31 03:00:00	4885.0	3	1	4	12	2002	365
2002-12-31 04:00:00	4857.0	4	1	4	12	2002	365
2002-12-31 05:00:00	4930.0	5	1	4	12	2002	365

In [11]:

```
fig, ax = plt.subplots(figsize=(10,8))
```

```
sns.boxplot(data=energy_df, x='month', y='PJMW_MW', palette='Blues')  
ax.set_title('MW by Month')  
plt.show()
```

Out [11]:

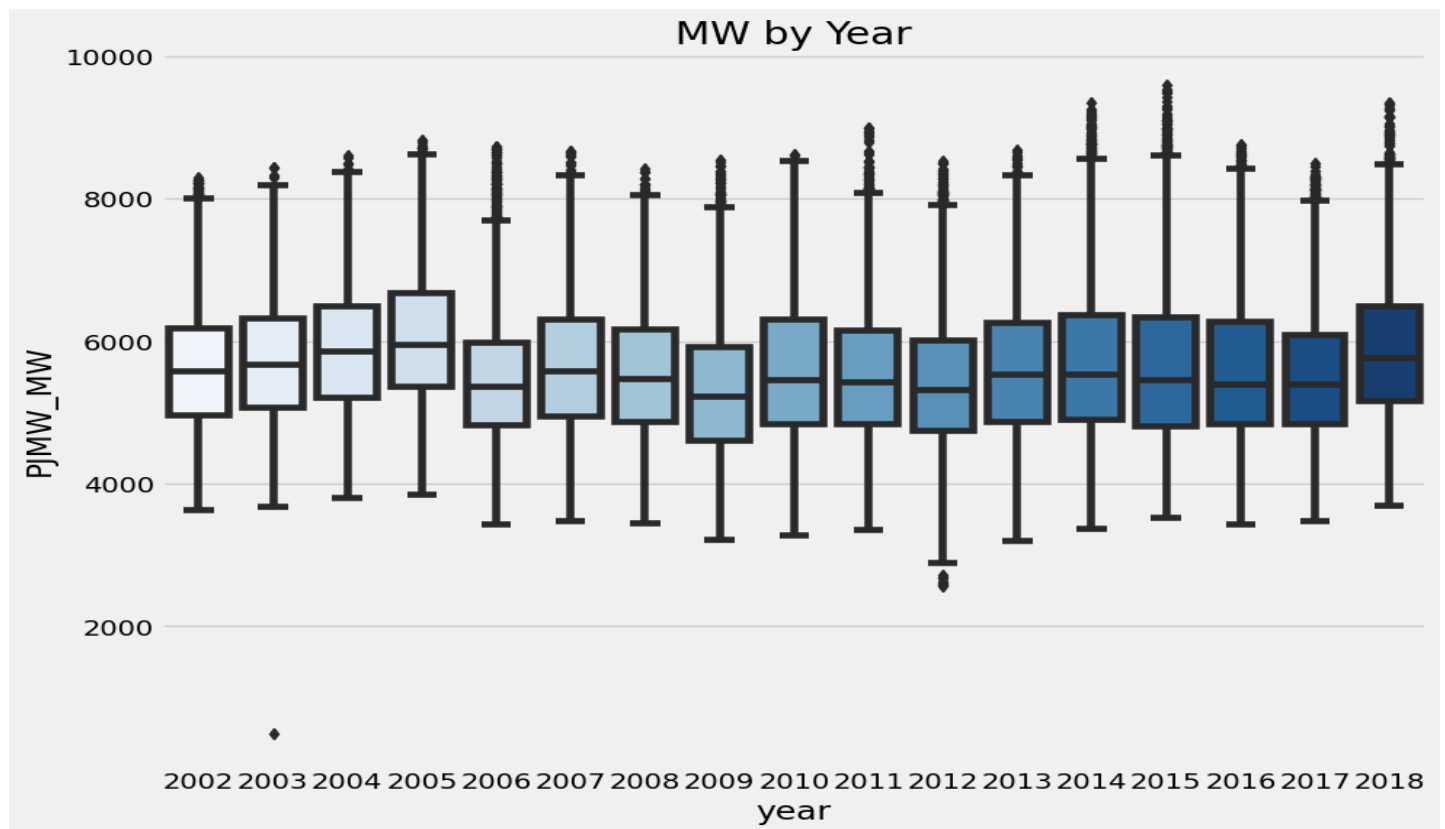


In [12]:

```
fig, ax = plt.subplots(figsize=(10,8))  
sns.boxplot(data=energy_df, x='year', y='PJMW_MW', palette='Blues')  
ax.set_title('MW by Year')  
plt.show()
```

#Visualize features and Target relationship

Out [12]:

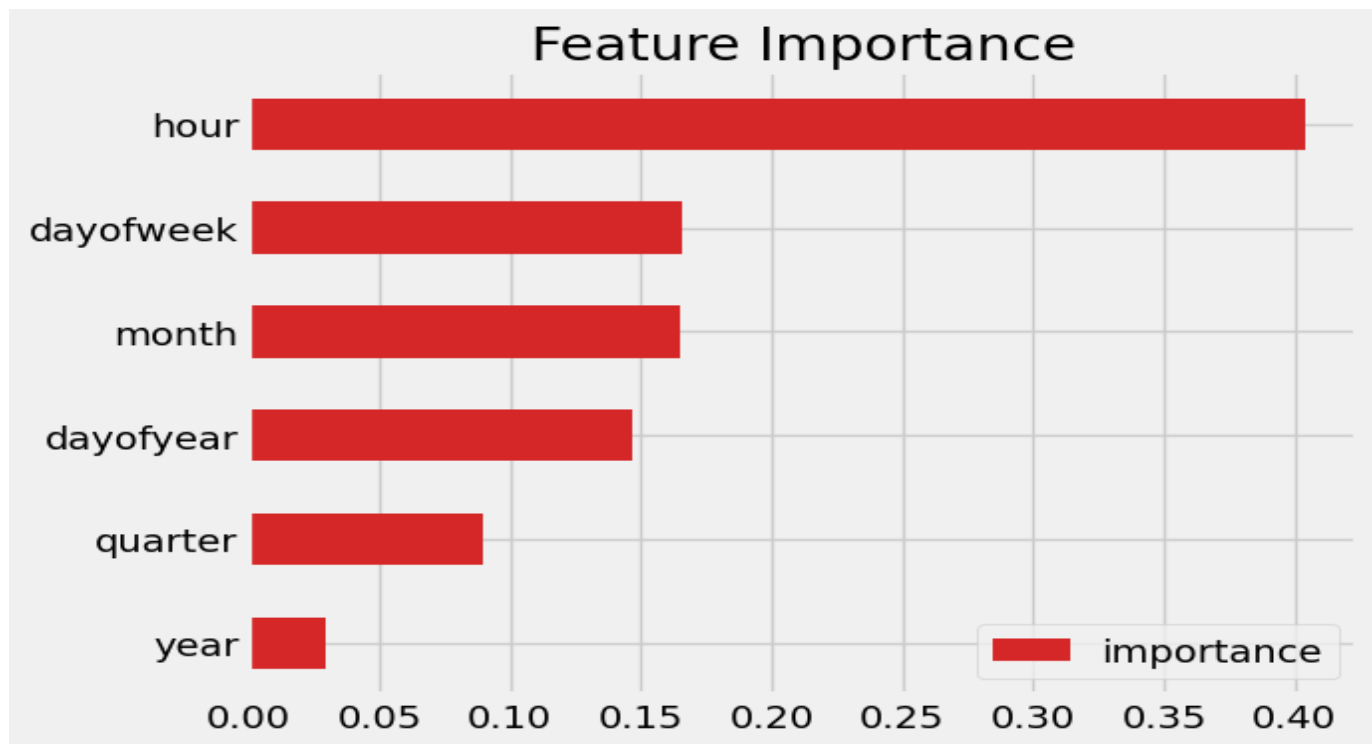


In [13]:

```
feature_imp.sort_values('importance').plot(kind='barh', title='Feature Importance', color=color_pal[3])  
plt.show()
```

# To determine feature importance, you can use various data analysis and machine learning techniques, such as linear regression, decision trees, random forests, or feature selection methods. These methods can help quantify the impact of each feature on energy consumption and identify which ones are the most influential in your specific context.

Out [13]:



**Some common data preprocessing tasks includes:-**

**1.Feature engineering:-**

This involves creating new features from the existing data. For example, this may involve creating features that represent interactions between variables, or features that represent summary statistics of the data.

**2.Data cleaning:-**

This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.



### **3.Data transformation:-**

This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.

### **4.Data integration:-**

This involves combining data from multiple sources into a single dataset. This may involve resolving inconsistencies in the data, such as different data formats or different variable name.

## **Conclusion & Future work:**

- Hourly energy consumption data analysis is crucial for understanding patterns and making informed decisions about energy management.
- PJM Energy consumption is sensitive to weather conditions. Hot summer days and cold winter days drive up electricity usage, emphasizing the need for robust demand response and forecasting capabilities to balance supply and demand.
- The conclusion is patterns can help identify areas where energy efficiency measures can be implemented, reducing energy wasted during low demand periods.
- The future work of this project is this project is a dynamic field with significant potential for improving energy efficiency, reducing costs.