MEASURE ENERGY CONSUMPTION

Phase-4: Development Part 2

Introduction:

Measuring energy consumption in Python is an essential practice for assessing and optimizing the efficiency of devices and systems. Python, as a versatile and powerful programming language, provides a flexible and capable platform for collecting, analyzing, and visualizing energy consumption data. Measuring energy consumption involves determining how much energy is used by a device, system, or an entire facility over a specific period. This is essential for various purposes, including tracking expenses, optimizing energy efficiency, and reducing environmental impact.

Content for Project Phase 4:

In this part you will continue building your project.

Continue the development by:

- ❖ Analyzing the energy consumption data.
- ❖ Creating visualizations.

Data Source:

- A good data source for measure energy consumption using machine learning and deep learning should be Accurate.
- The dataset used in this project is obtained from Kaggle.

Dataset Link:

https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption

Datetime	PJMW_MW
31-12-2002 01:00	5077
31-12-2002 02:00	4939
31-12-2002 03:00	4885
31-12-2002 04:00	4857
31-12-2002 05:00	4930
31-12-2002 06:00	5126
31-12-2002 07:00	5493
31-12-2002 08:00	5824
31-12-2002 09:00	5962
31-12-2002 10:00	6019
31-12-2002 11:00	5988
31-12-2002 12:00	5885
31-12-2002 13:00	5764
31-12-2002 14:00	5612
31-12-2002 15:00	5486
31-12-2002 16:00	5474
31-12-2002 17:00	5668
31-12-2002 18:00	6027
31-12-2002 19:00	5973

Data Analyzing:

Analyzing data in Python refers to the process of examining and extracting meaningful insights, patterns, and information from a dataset using Python programming and data analysis libraries. This process can include various tasks such as data exploration, data cleaning, statistical analysis, visualization, and machine learning. Here are some key aspects of analyzing data in Python:

Import the libraries:

```
import pandas as pd
importnumpy as np
importmatplotlib.pyplot as plt
importseaborn as sns
```

Data Loading:

The first step is to import your dataset into Python. You can use libraries like pandas to load data from various sources, including CSV files, Excel spreadsheets, databases, and more.

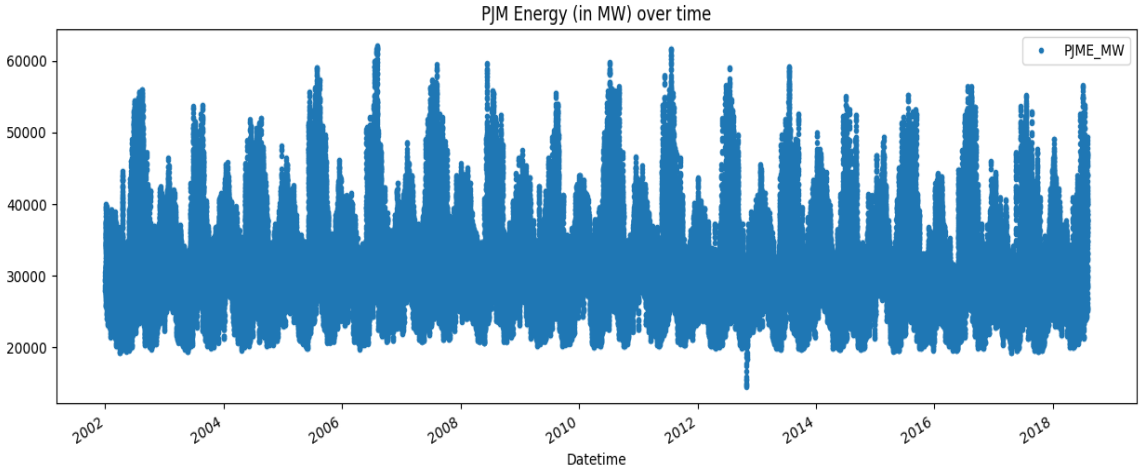
```
data = pd.read_csv('PJMW_hourly.csv')
```

Data Exploration:

This involves getting to know your dataset, including its structure, column names, and initial summary statistics. You'll often start by using functions like df.head() and df.info() to understand the data's basic characteristics.

```
# turn data to datetime
df = df.set_index('Datetime')
df.index = pd.to_datetime(df.index)
```

```
# create the plot
df.plot(style='.', figsize=(15, 5), title='PJM Energy (in MW) over time')
plt.show()
```

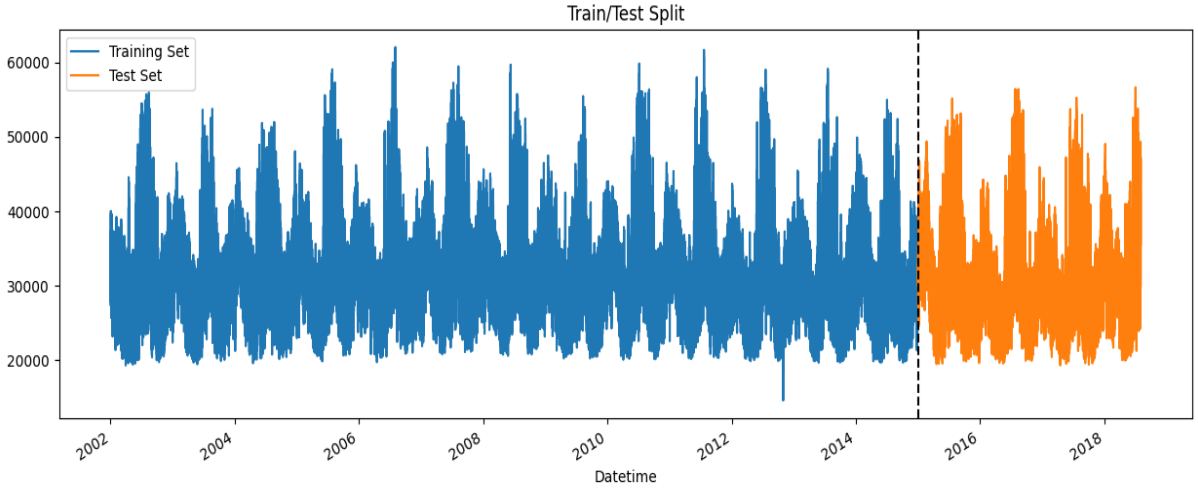


Splitting data:

Splitting data in the context of measuring energy consumption typically refers to the practice of dividing a dataset into two or more subsets for various purposes, such as training and testing machine learning models, validating the model's performance, or conducting statistical analyses.

```
# train / test split
train = df.loc[df.index< '01-01-2015']
test = df.loc[df.index>= '01-01-2015']

fig, ax = plt.subplots(figsize=(15, 5))

train.plot(ax=ax, label='Training Set', title='Train/Test Split')

test.plot(ax=ax, label='Test Set')

ax.axvline('01-01-2015', color='black', ls='--')
```

```
ax.legend(['Training Set', 'Test Set'])
plt.show()
```



Data Cleaning:

Data cleaning is crucial to deal with missing values, duplicates, and inconsistent data. You can use Pandas to perform tasks like removing or imputing missing values, dropping duplicates, and standardizing data types.

Which involves:

1. Handling Missing Data
2. Dealing with Duplicates
3. Data Type Conversion
4. Outlier Detection

```
data['Year'] = data['Datetime'].dt.year
data['Month'] = data['Datetime'].dt.month
data['Day'] = data['Datetime'].dt.day
data['Hour'] = data['Datetime'].dt.hour
```

Data Transformation:

Data might need to be transformed for analysis. This includes creating new features, encoding categorical variables, or normalizing numerical data.

```
data['Date'] = pd.to_datetime(data['Date'])
```

Preprocessing:

Preprocessing data in Python is the process of cleaning, transforming, and organizing raw data to make it suitable for analysis, visualization, or machine learning. Proper data preprocessing is essential to ensure the accuracy and reliability of your analysis.

```
train = create_features(train)
test = create_features(test)
features = ['dayofyear', 'hour', 'dayofweek', 'quarter',
'month', 'year'] target = 'PJME_MW'
X_train = train[features]
y_train = train[target]
X_test = test[features]
y_test = test[target]
```

Data Visualizing:

Data visualization is the process of representing data in graphical or visual formats, such as charts, graphs, maps, and other visual elements. The goal of data visualization is to make complex data more understandable, accessible, and interpretable.

It allows individuals to quickly grasp insights, patterns, and trends within the data, which might be challenging to discern from raw data or textual descriptions alone.

It is a fundamental part of data analysis and reporting, enabling analysts, data scientists, and decision-makers to better understand data, make informed decisions, and communicate their findings effectively.

There are numerous data visualization tools and libraries available, such as Matplotlib, Seaborn, Tableau, and D3.js, which cater to a wide range of visualization needs.

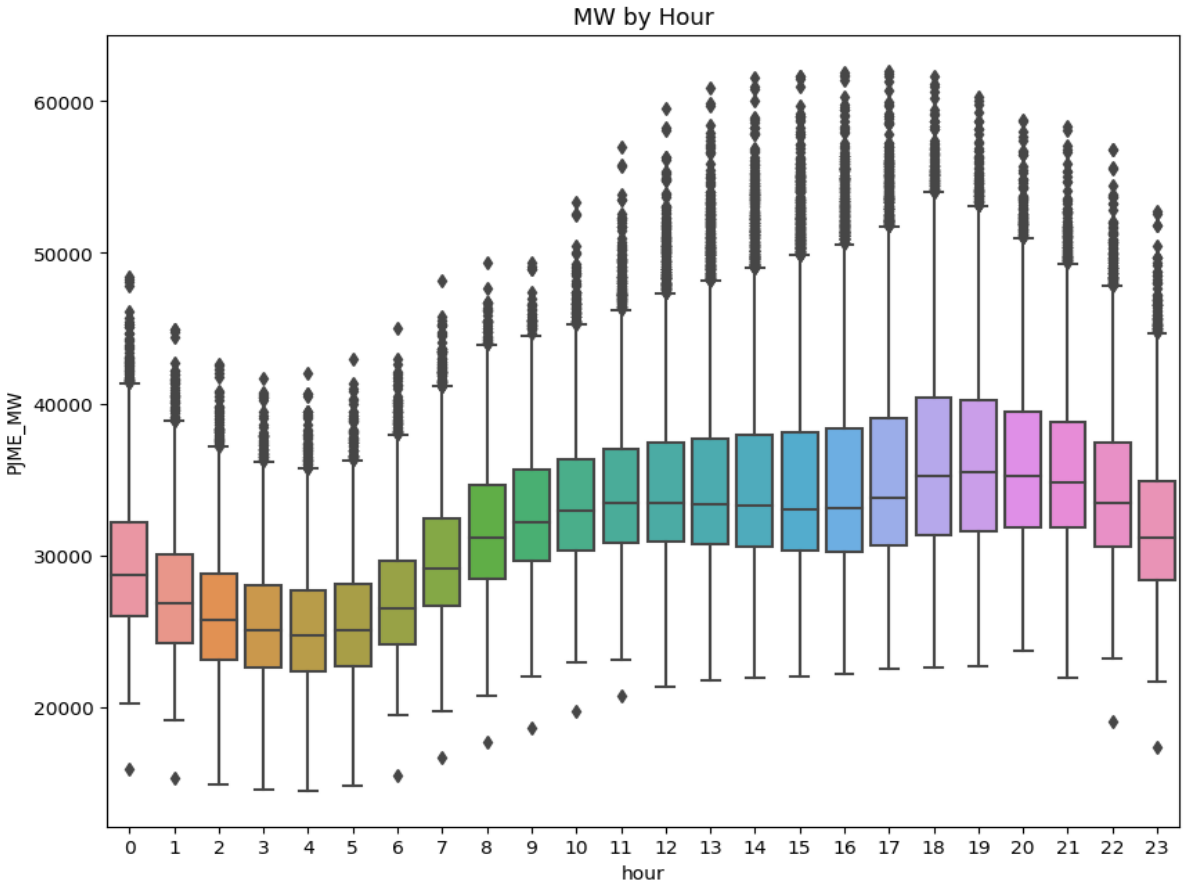
Create visualizations to present energy consumption trends and insights. This can include:

- Time series plots to visualize changes over time.
- Histograms or bar charts to show distribution of energy consumption.
- Heatmaps to display correlations between features.
- Box plots or violin plots for outlier detection.
- Geographic maps to visualize regional variations.
- Use visualization libraries like Matplotlib, Seaborn, Plotly, or geographic mapping tools like Folium (for maps).

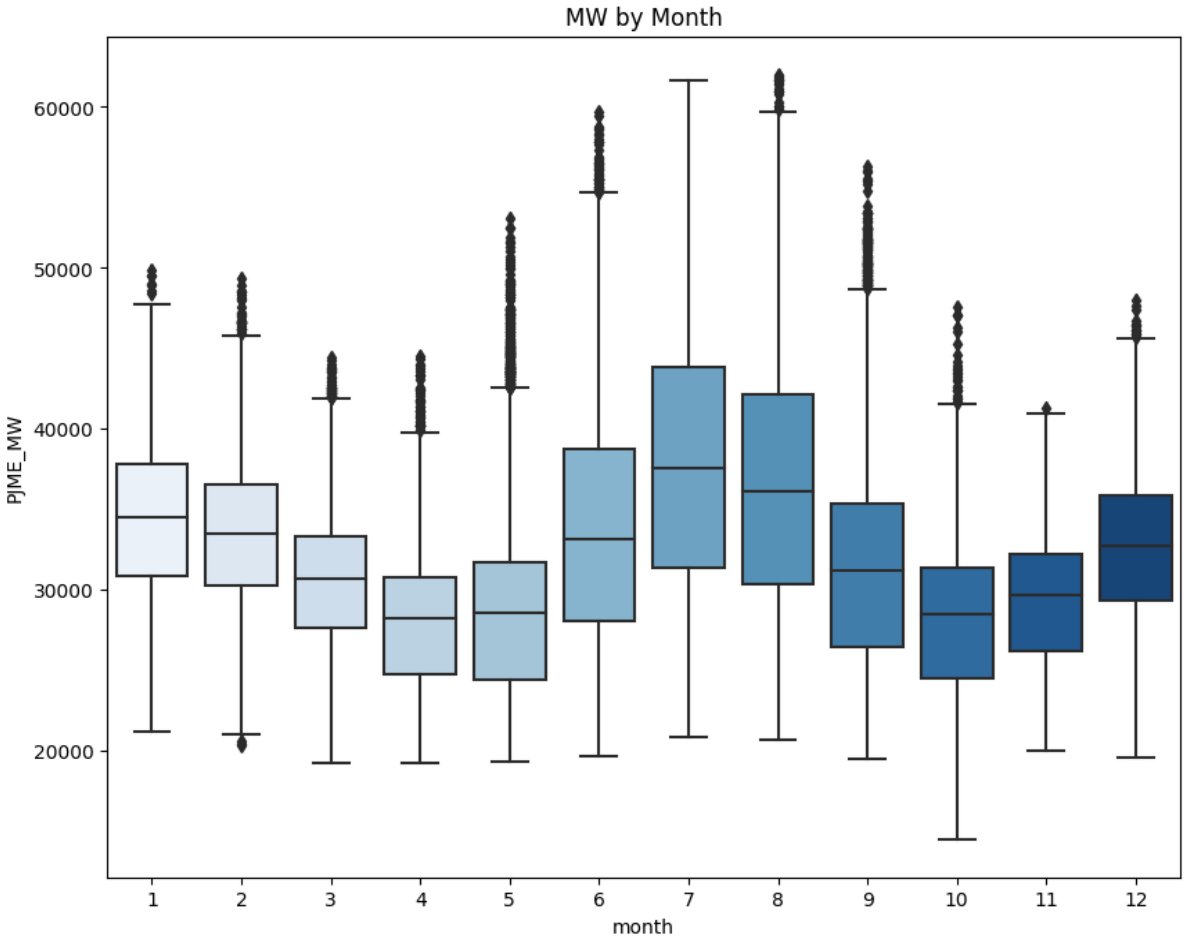
Key aspects of data visualization include:

- Graphical Representations.
- Simplification.
- Pattern Recognition.
- Time Series Visualization.
- Exploratory Data Analysis (EDA).
- Big Data Visualization.
- Geospatial Visualization.
- Scientific Visualization.

```
# visualize the hourly Megawatt
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='hour', y='PJME_MW')
ax.set_title('MW by Hour')
plt.show()
```




```
# viaualize the monthly Megawatt
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='month', y='PJME_MW',
palette='Blues')
ax.set_title('MW by Month')
plt.show()
```


MW by Month
PJME_MW
60000
50000
40000
30000
20000
1
2
3
4
5
6
7
8
9
10
11
12
month


Thank You!