**MEASURE ENERGY CONSUMPTION**

**422521104702: SALMAAN SHAREEF.M**

**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.

**Content for Project Phase 4:**

In this part you will continue building your project.

Continue the development by:

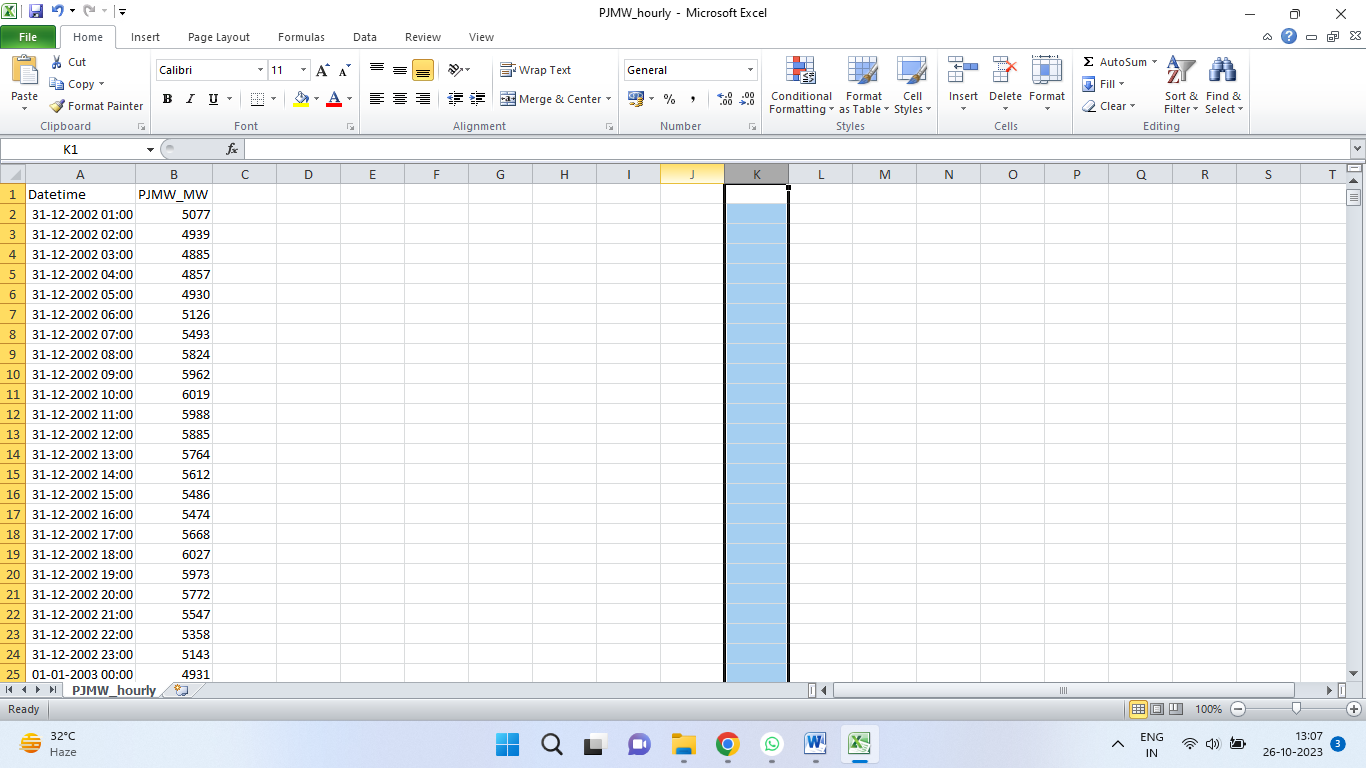
* Analyzing the energy consumption data.
* Creating visualizations.

**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**](https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption)

****

**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:

* **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 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.

* **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 Transformation:**

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

* **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.

**Data Visualizing:**

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).

**PROGRAM:**

# Import the libraries:

# import pandas as pd

# import numpy as np

# import matplotlib.pyplot as plt

# import seaborn as sns

# Load the Dataset:

# data = pd.read\_csv('PJMW\_hourly.csv')

# Explore the data:

# # 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()

# Split the data:

# # 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))

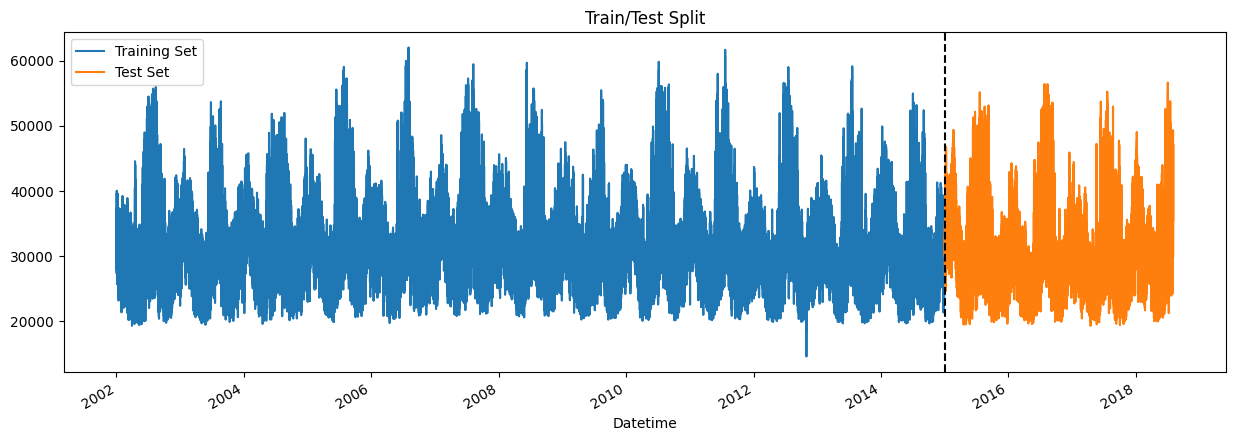
train.plot(ax=ax, label='Training Set', title='Train/Test Split')

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

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

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

plt.show()



# Data Transformation:

data['Date'] = pd.to\_datetime(data['Date'])

# Data Preprocessing:

# data['Year'] = data['Datetime'].dt.year

# data['Month'] = data['Datetime'].dt.month

# data['Day'] = data['Datetime'].dt.day

# data['Hour'] = data['Datetime'].dt.hour

# 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]

# Visualizing data:

# # 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()

# 

