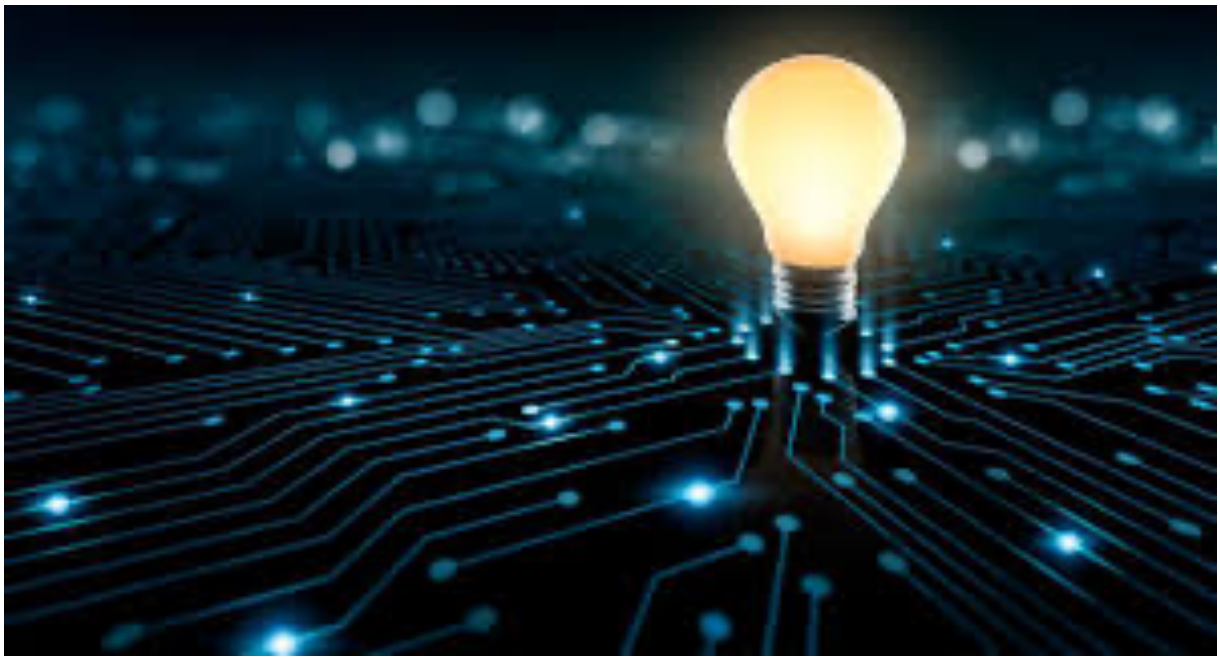
MEASURE ENERGY CONSUMPTION

Team Leader

422521104016 : KAMALESH P

Phase-3 Documentation Submission



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. In this context, we'll explore the fundamental concepts and tools involved in measuring and managing energy consumption using Python, empowering you to make data-driven decisions and contribute to sustainability efforts.

Content for Project Phase 3:

- In this part you will begin building your project by loading and preprocessing the dataset.

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

Data Loading :

Load the energy consumption dataset, which can be in various formats such as CSV, Excel, or a database. You can use libraries like pandas, numpy, or sqlite3, depending on the data format.

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.

PROGRAM:

Import the libraries:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns
```

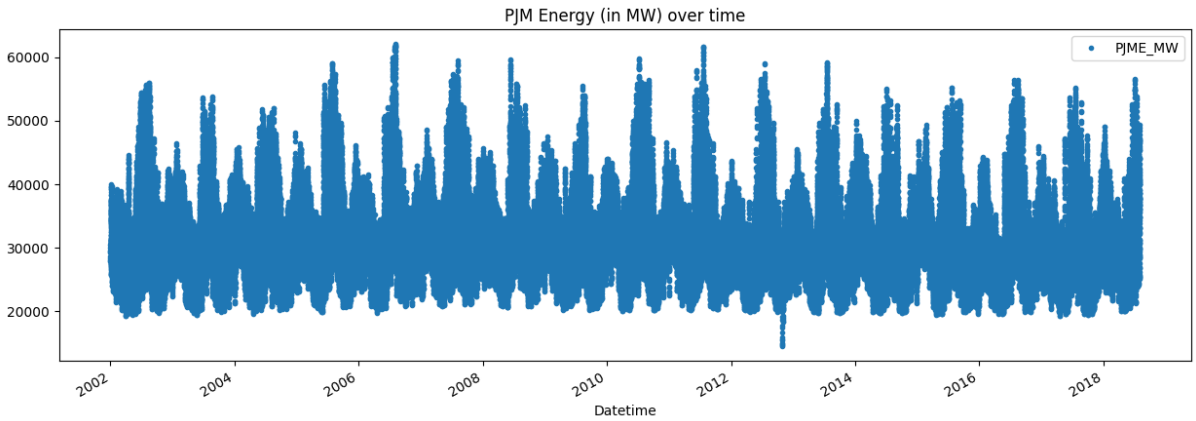
Load the data:

```
df = 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()
```

Preprocessing:

Handling Missing Values:

- Check for missing data, as missing values can affect the quality of your analysis. Depending on the extent of missing data, you can either drop rows with missing values or impute the missing values using techniques like mean, median, or interpolation.

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

print("Missing Values:")

print(missing_values)
```

Data Transformation

- Convert date and time columns to datetime objects if your dataset includes timestamps.
- Convert categorical variables into numerical representations if needed, using techniques like one-hot encoding or label encoding.

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

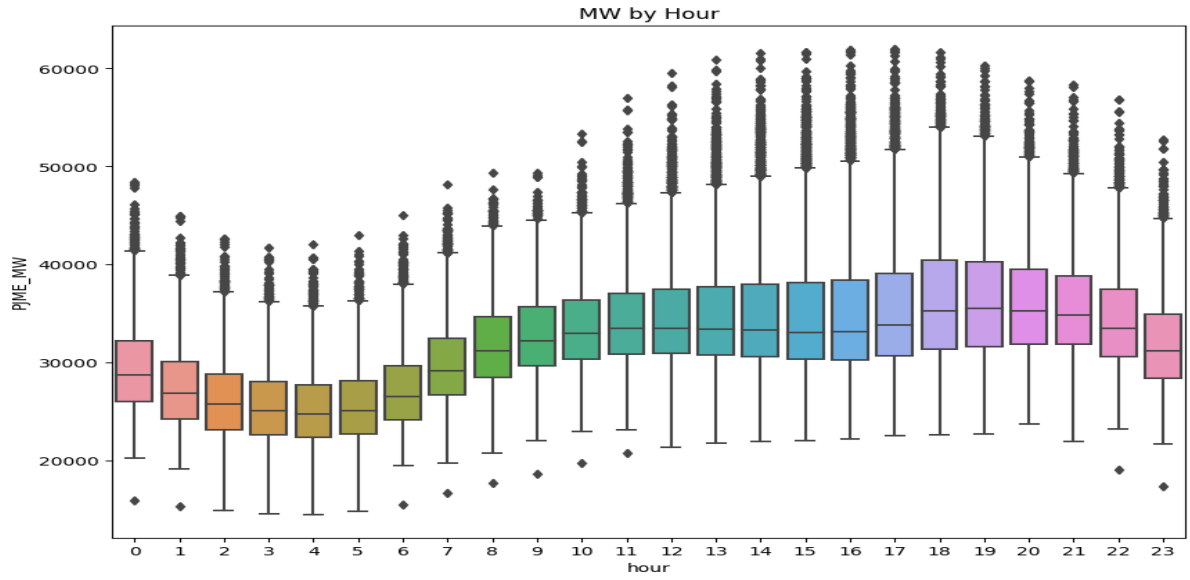
Feature Engineering:

- Extract relevant features from the data, such as day of the week, hour of the day, or season, which can be important for energy consumption analysis.
- Create lag features, which involve using past values of energy consumption to predict future values.

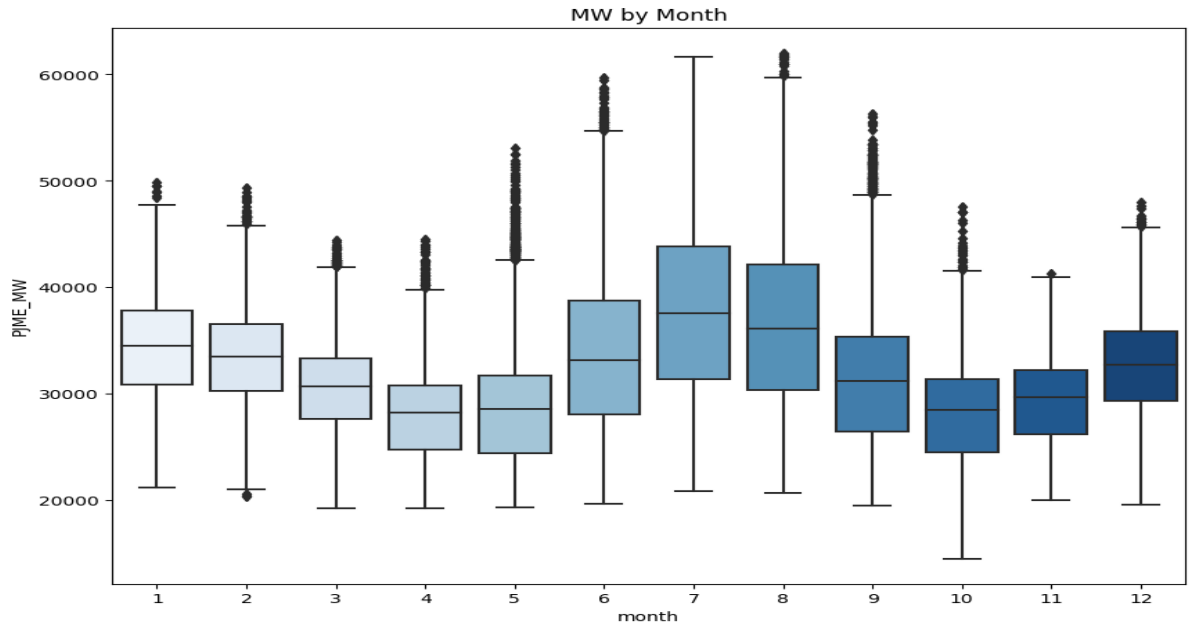
```
# feature creation
def create_features(df):
    df = df.copy()
    df['hour'] = df.index.hour
    df['dayofweek'] = df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] = df.index.dayofyear
    df['dayofmonth'] = df.index.day
    df['weekofyear'] = df.index.isocalendar().week
    return df
df = create_features(df)
```

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()
```



Data Scaling:

- Scale numerical features, especially if you're planning to use machine learning algorithms. Common scaling methods include Min-Max scaling (normalization) and standardization.

```
scaler = MinMaxScaler()

data[['EnergyConsumption', 'Temperature']] =
scaler.fit_transform(data[['EnergyConsumption', 'Temperature']])
```

Conclusion:

- Measuring energy consumption using Python is an essential endeavor, and it begins with the meticulous process of data loading and preprocessing.
- Python's data manipulation libraries, particularly Pandas, are instrumental in efficiently loading diverse data sources, from CSV to databases. Through careful preprocessing, we clean, transform, and structure the data, ensuring its integrity and relevance. This entails addressing missing values, encoding categorical variables, and scaling features when necessary.
- The combination of data loading and preprocessing sets the foundation for robust energy consumption analysis and modeling. It empowers us to uncover insights and relationships, paving the way for data-informed strategies that optimize energy use, reduce waste, and contribute to sustainable energy management practices.
- Python's capabilities in this regard make it an invaluable tool for the vital task of energy consumption measurement and analysis.


Thank You!