Reg no :721921243053

Name: Jebin Alias

Project: ***Measure Energy Consumption***

Introduction:

- Energy consumption in the body is a product of the basal metabolic rate and the physical activity level.
- The physical activity level are defined for a non-pregnant, non-lactating adult as that person's total energy expenditure (TEE) in a 24-hour period, divided by his or her basal metabolic rate (BMR).
- We will focus on analyzing energy consumption data and creating visualizations based on the findings.

Objective:

- Initiate the development process by selecting a suitable dataset and preparing it for analysis.
- In PHASE 1 we discussed about the problem definition and their application used in artificial intelligence.
- In phase 2 we discussed about the explore innovative technologies used in artificial intelligence.
- In phase 3 we discussed the development process for energy consumption >

***PHASE 4*:**

- To continue the development by analyzing the energy consumption data and creating visualization
- In this part you will continue building your project.
- Continue the development by:

  1) Analyzing the energy consumption data

  2) Creating visualizations.

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

Analyzing Energy Consumption Data

you'll need to follow these steps such as,

1. Data Cleaning and Preparation:

- Load the energy consumption data into a suitable data structure (e.g., Data Frame if using Python).

- Check for missing values, outliers, and any anomalies in the data.

- Handle any data quality issues by imputing missing values or removing outliers.

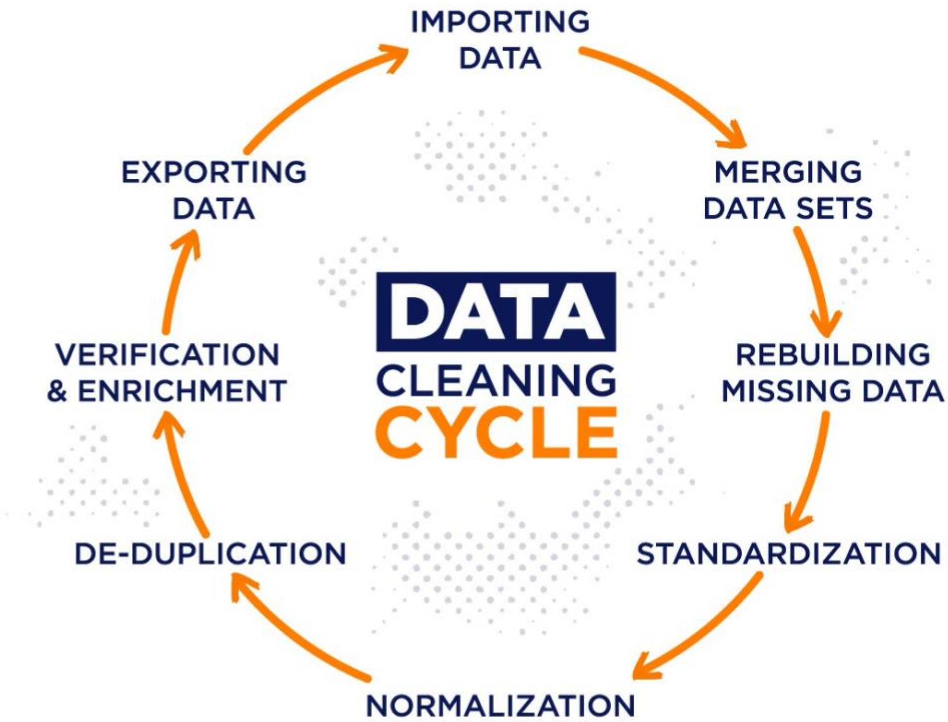


Fig.1.1 data cleaning cycle

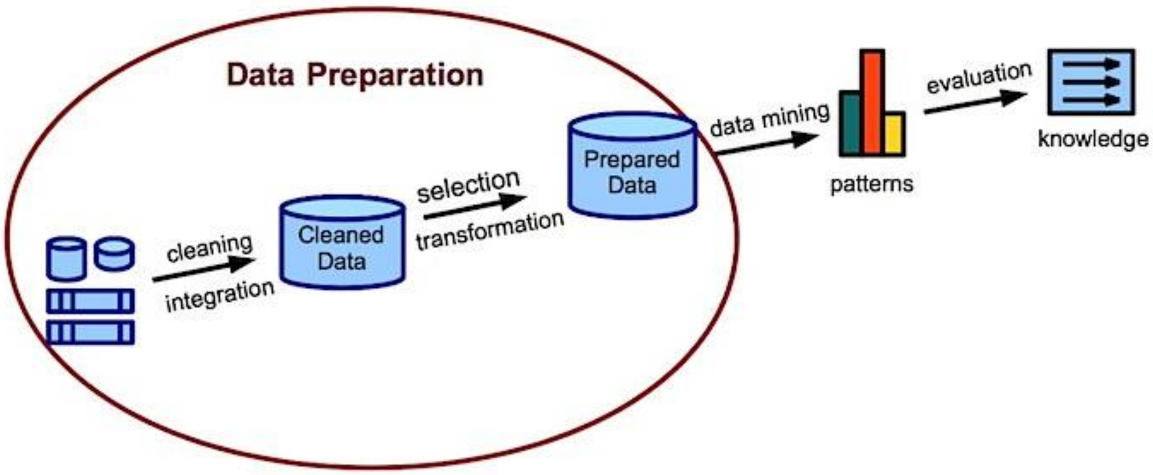


Fig.1.2 Data preparation

2. Exploratory Data Analysis (EDA):

- Calculate basic statistics (mean, median, standard deviation, etc.) to understand the central tendency and variability of energy consumption.

- Visualize the distribution of energy consumption using histograms, box plots, or violin plots.

- Identify any trends or patterns in the data over time.

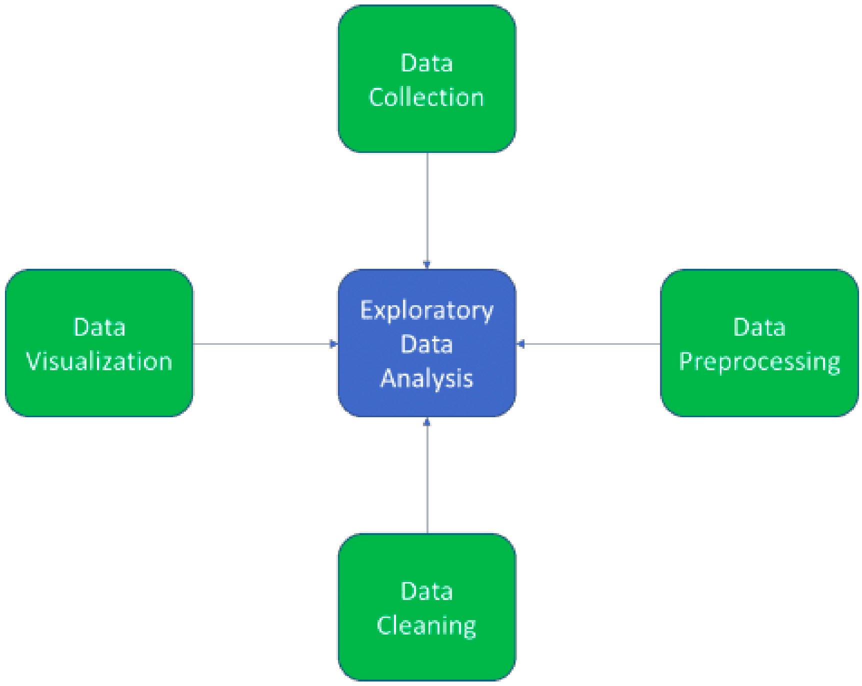


Fig.1.3 Exploratory data analysis

3. Time Series Analysis :

- If the data is time-stamped, perform time series analysis to uncover seasonality, trends, and cyclic patterns.

- Apply techniques like decomposition, autocorrelation, and rolling statistics to gain insights.

Components :

1. Trend
2. Cyclical variation
3. Random or Irregular movements
4. Seasonal variations

Types:

i. **Classification:** It picks out and assigns categories or segments to the data**.**
ii. **Descriptive Analysis:** It involves identifying the patterns in the time series data.
iii. **Curve Fitting:** It plots the data on a curve for investigating the relationships among variables within the data.
iv. **Segmentation:** It splits the data into different categories to disclose the source data's underlying properties.

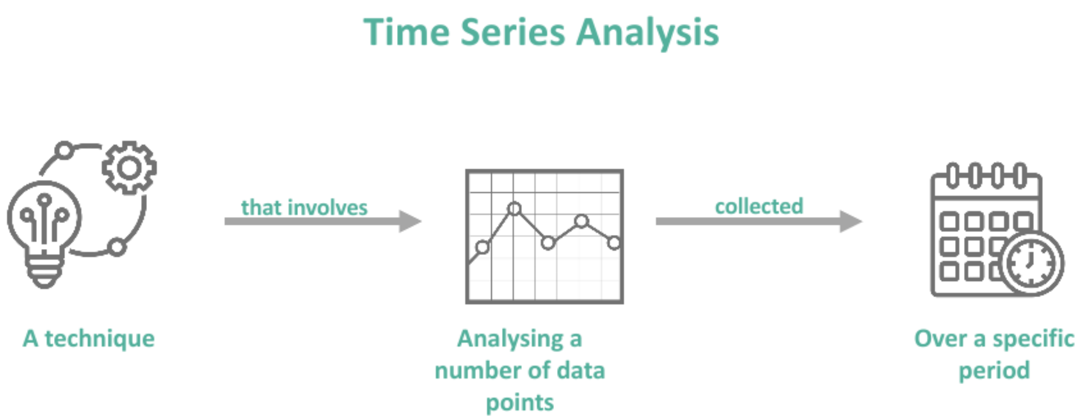


Fig.1.4 Time series analysis

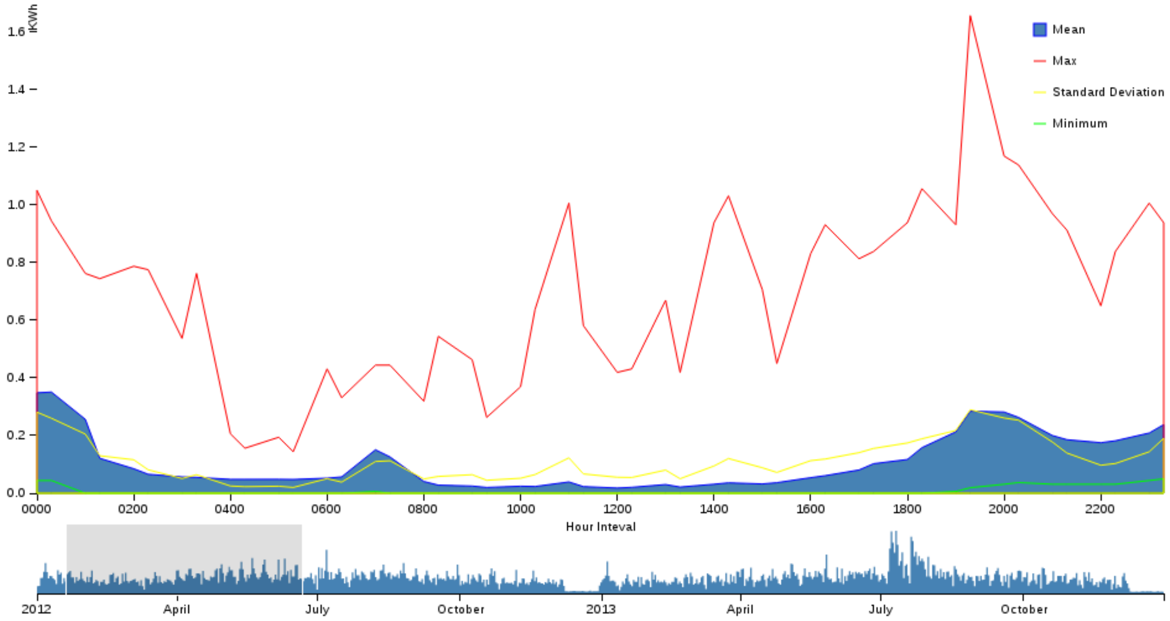
Creating Visualizations

1. Energy Consumption Trends Over Time:

- Generate a line chart to visualize how energy consumption has evolved over the specified time period.

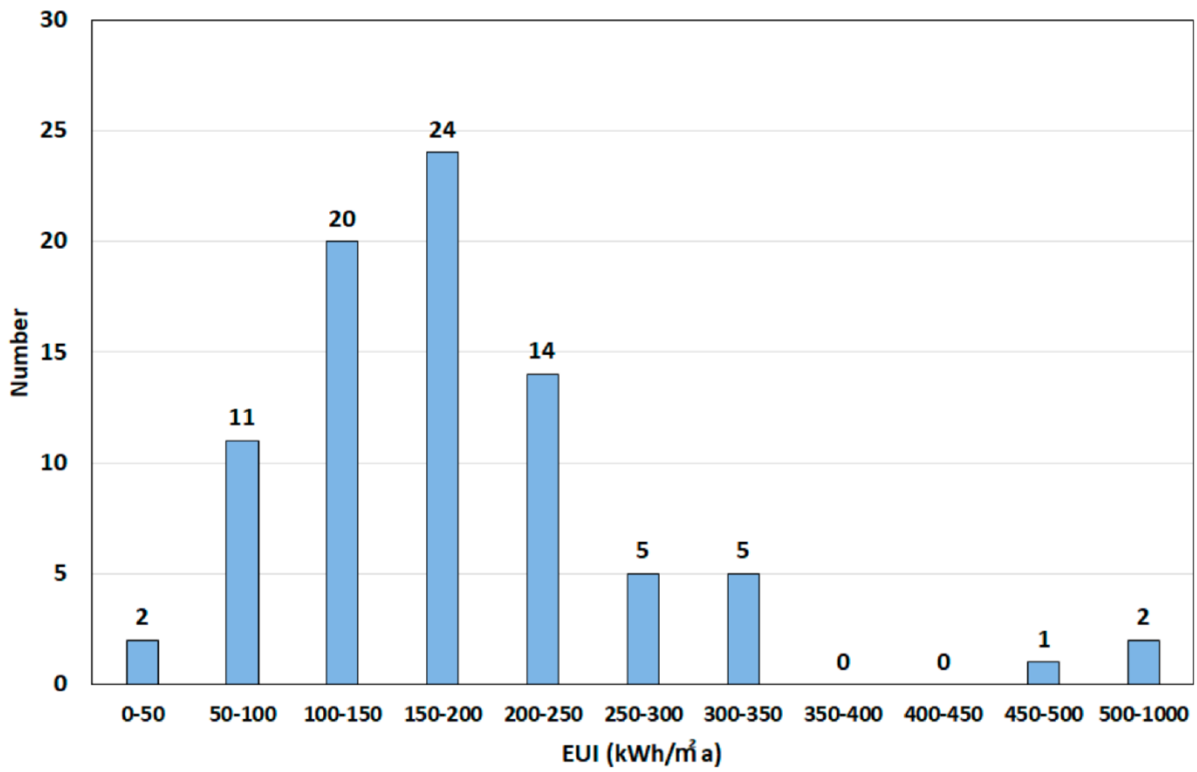
- Add annotations or markers to highlight significant events or changes.

Visualising Energy Consumption Profile (by hour of day)



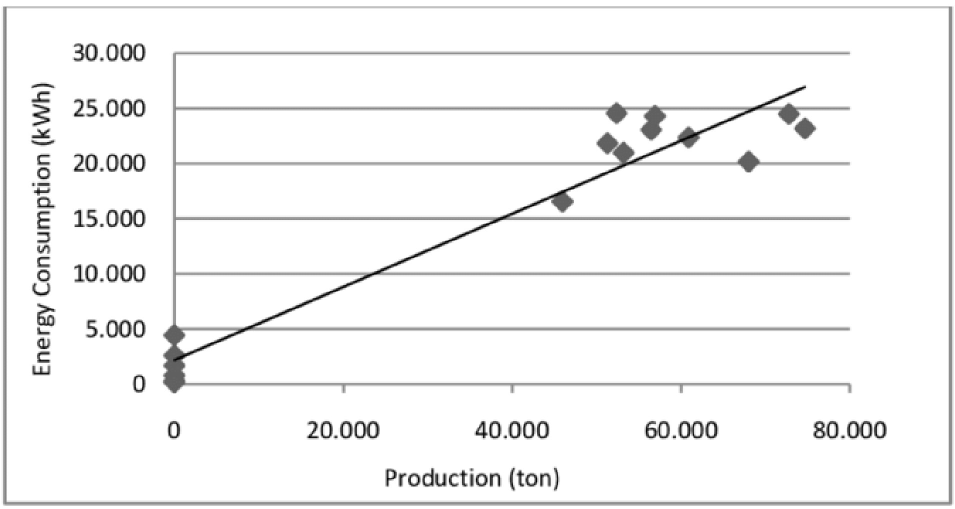
2. Comparative Analysis:

- Create bar charts or stacked area charts to compare energy consumption across different categories, regions, or units (if applicable).



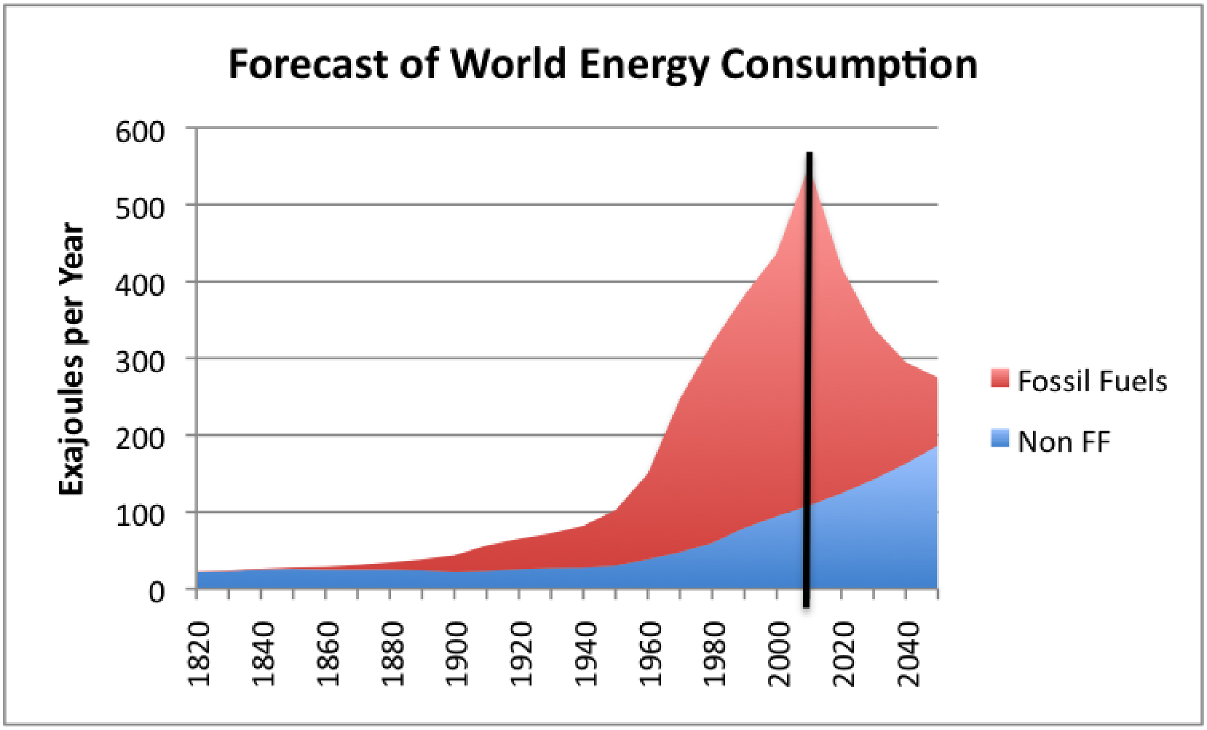
3. Correlation Analysis:

- Generate scatter plots or heatmaps to visualize the relationships between energy consumption and other relevant variables (e.g., temperature, occupancy, etc.).



4. Forecasting :

- If forecasting is a goal, create visualizations that display actual vs. predicted energy consumption.



5. Geospatial Visualization :

- If the data contains location information, create maps to visualize energy consumption distribution across different regions.

Remember to label your visualizations appropriately, provide legends or color keys, and add titles and captions to convey the insights effectively.

<u>Documentation and Reporting</u>

Lastly, document your analysis and visualization process. This should include:

- A summary of the key findings and insights.

- Any assumptions or data preprocessing steps taken.

- Interpretation of the visualizations and what they reveal about energy consumption patterns.

- Recommendations or actions based on the insights gained.

By following these steps, you'll be able to effectively analyze the energy consumption data and create meaningful visualizations to communicate your findings.

Source code:

Int[1]:

```
Import numpy as np
Import pandas as pd
Import matplotlib.pyplot as plt
Import matplotlib.dates as mdates
%matplotlib inline
Import seaborn as sns
Import warnings
Warnings.filterwarnings(“ignore”)
```

From pandas.plotting import lag_plot

From pylab import rcParams

From statsmodels.tsa.seasonal import seasonal_decompose

From pandas import DataFrame

From pandas import concat

Int[2]:

Df=pd.read_csv("../input/hourly-energy-consumption/AEP_hourly.csv",index_col='Datetime',parse_dates=True)

Df.head()

Out[2]:

	AEP_MW
Datetime	
2004-12-31 01:00:00	13478.0
2004-12-31 02:00:00	12865.0
2004-12-31 03:00:00	12577.0
2004-12-31 04:00:00	12517.0
2004-12-31 05:00:00	12670.0

Int[3]:

```
 df.sort_values(by='Datetime', inplace=True)
print(df)
```

Int[4]:

```
 df.shape
```

Out[4]:

```
(121273, 1)
```

Int[5]:

```
df.info()
```

Out[5]:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 121273 entries, 2004-10-01 01:00:00 to 2018-08-03 00:00:00
Data columns (total 1 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   AEP_MW  121273 non-null  float64
dtypes: float64(1)
memory usage: 1.9 MB
```

Int[6]:

```
df.describe()
```

Out[6]:

	AEP_MW
count	121273.000000
mean	15499.513717
std	2591.399065
min	9581.000000
25%	13630.000000
50%	15310.000000
75%	17200.000000
100%	25695.000000

Int[7]:

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

Int[8]:

```
# Extract all Data Like Year MOnth Day Time etc
df["Month"] = df.index.month
```

```
df["Year"] = df.index.year
df["Date"] = df.index.date
df["Hour"] = df.index.hour
df["Week"] = df.index.week
df["Day"] = df.index.day_name()
df.head()
```

Out[8]:

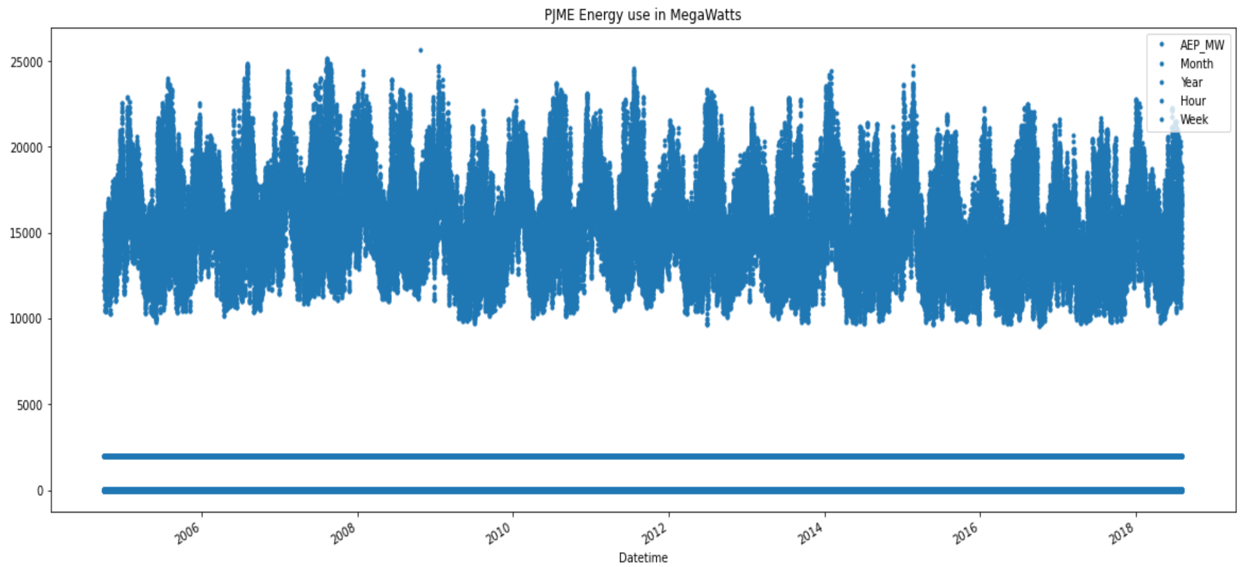
	AEP_MW	Month	Year	Date	Hour	Week	Day
Datetime							
2004-10-01 01:00:00	12379.0	10	2004	2004-10-01 1	1	4	Friday
2004-10-01 02:0	11935.0	10	2004	2004-10-01 1	2	4	Friday

0:00							
2004-10-01 03:00:00	11692.0	10	2004	2004-10-01 1	3	4	Friday
2004-10-01 04:00:00	11597.0	10	2004	2004-10-01 1	4	4	Friday
2004-10-01 05:00:00	05:00:00	10	2004	2004-10-01 1	5	4	Friday

Int[9]:

```
df.plot(title="PJME Energy use in MegaWatts",
        figsize=(20, 8),
        style=".",
        color=sns.color_palette()[0])

plt.show()
```

Out[9]:



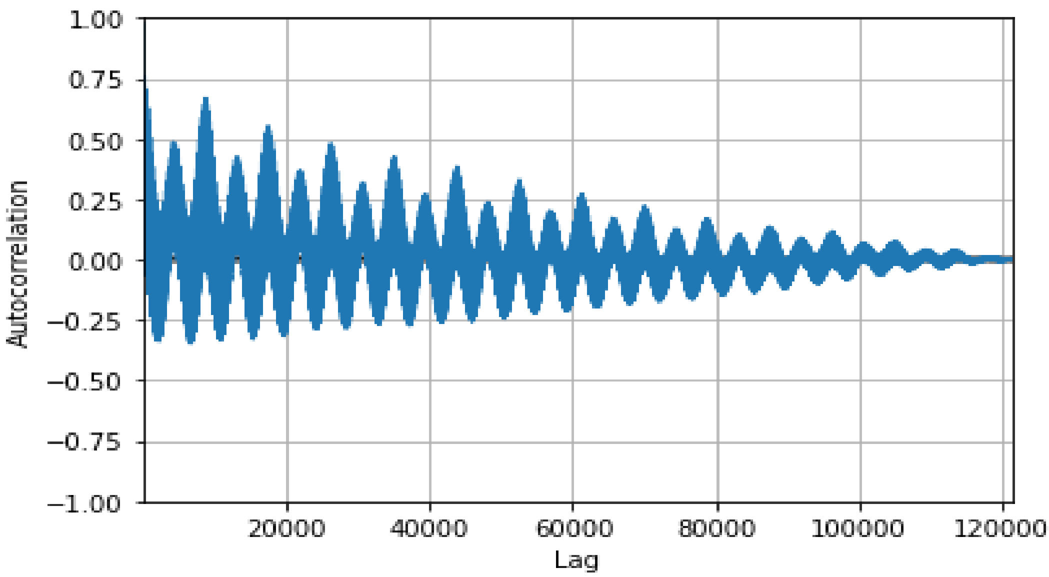
Int[10]:

from pandas.plotting import autocorrelation_plot

autocorrelation_plot(df['AEP_MW'])

plt.show()

Out[10]:



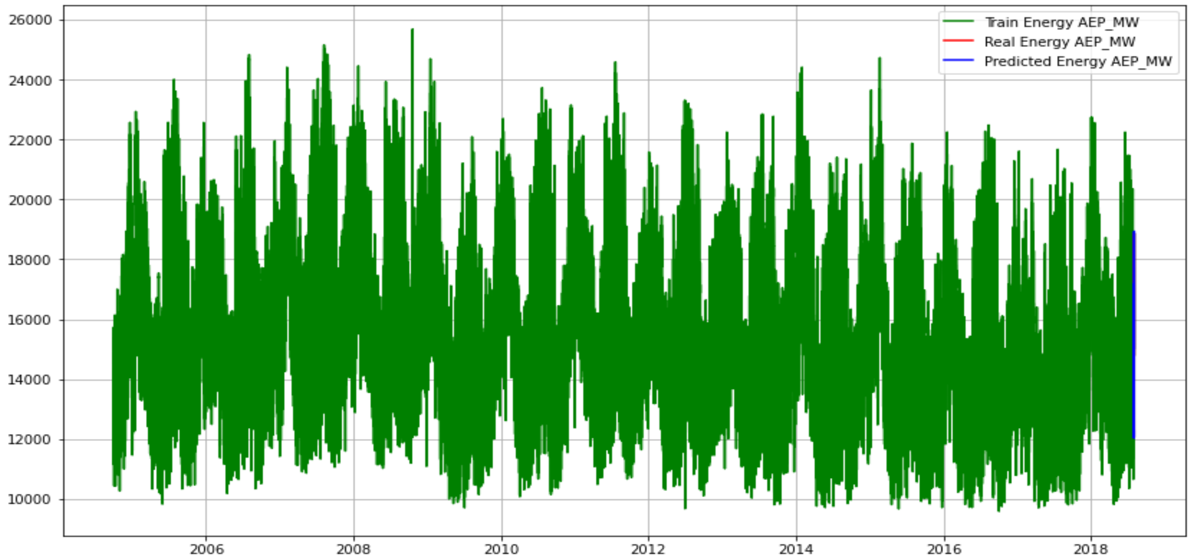
Int[11]:

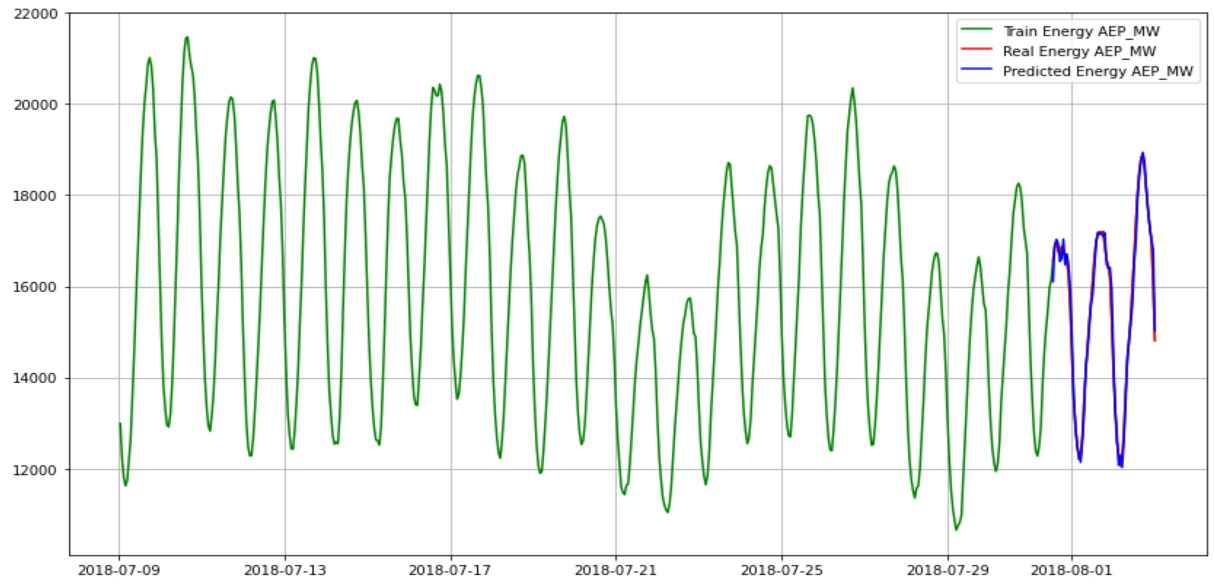
```
#Train Arima Model
train_arima = train_data['AEP_MW']
test_arima = test_data['AEP_MW']

history = [x for x in train_arima]
y = test_arima
# make first prediction
predictions = list()
model = sm.tsa.arima.ARIMA(history,
order=(5,1,0))
model_fit = model.fit()
yhat = model_fit.forecast()[0]
predictions.append(yhat)
history.append(y[0])
# rolling forecasts
for i in range(1, len(y)):
    # predict
    model = sm.tsa.arima.ARIMA(history,
order=(5,1,0))
    model_fit = model.fit()
    yhat = model_fit.forecast()[0]
```

```
    # invert transformed prediction
    predictions.append(yhat)
    # observation
    obs = y[i]
    history.append(obs)

plt.figure(figsize=(14,8))
plt.plot(df.index, df['AEP_MW'], color='green',
label = 'Train Energy AEP_MW')
plt.plot(test_data.index, y, color = 'red', label = 'Real
Energy AEP_MW')
plt.plot(test_data.index, predictions, color = 'blue',
label = 'Predicted Energy AEP_MW')
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(14,8))
plt.plot(df.index[-600:], df['AEP_MW'].tail(600),
color='green', label = 'Train Energy AEP_MW')
plt.plot(test_data.index, y, color = 'red', label = 'Real
Energy AEP_MW')
```

```
plt.plot(test_data.index, predictions, color = 'blue', label = 'Predicted Energy AEP_MW')
plt.legend()
plt.grid(True)
plt.show()

print('MSE: '+str(mean_squared_error(y, predictions)))
print('MAE: '+str(mean_absolute_error(y, predictions)))
print('RMSE: '+str(sqrt(mean_squared_error(y, predictions))))
```

Out[11]:




22000
20000
18000
16000
14000
12000
2018-07-09
2018-07-13
2018-07-17
2018-07-21
2018-07-25
2018-07-29
2018-08-01
Train Energy AEP_MW
Real Energy AEP_MW
Predicted Energy AEP_MW

Conclusion:

- Remember to iterate through these steps as necessary, and always validate your findings with domain experts if possible.

- Regularly updating the model with new data is crucial to ensure it remains accurate as consumption patterns evolve.