Measure Energy Consumption

Date	18 October 2023
Team ID	Proj_212174_Team_1
Project Name	Measurement of Energy consumption
Maximum Marks	

Data Visualization:

Data visualization is the presentation of data in graphical or pictorial format to help people understand the information more easily. It involves creating charts, graphs, maps, and other visual elements to represent data points, patterns, and trends. Data visualization is a powerful tool for making complex data more accessible and can be used in various fields, including business, science, and journalism, to make data-driven decisions and communicate insights effectively.

Matplotlib:

Matplotlib is a popular Python library used for creating static, animated, and interactive visualizations. It provides a wide range of functions and classes for generating various types of 2D and 3D plots, charts, and graphs. Matplotlib is highly customizable, allowing users to control almost every aspect of their visualizations, such as colors, labels, and styles.

It is commonly used for tasks like creating line plots, scatter plots, bar charts, histograms, and more. Matplotlib is often used in conjunction with other Python libraries like NumPy and Pandas for data manipulation and analysis. This library is widely employed in scientific research, data analysis, and data visualization projects.

Seaborn:

Seaborn is a Python data visualization library built on top of Matplotlib. It provides a higher-level interface for creating attractive and informative statistical graphics. Seaborn is particularly well-suited for working with complex datasets and visualizing relationships between variables.

Some of its key features include:

1. Improved Aesthetics: Seaborn comes with a variety of built-in themes and color palettes to make plots more visually appealing and easy to read.

2. High-Level Interface: It simplifies the process of creating complex statistical plots, allowing users to create attractive visualizations with minimal code.

3. Integration with Pandas: Seaborn seamlessly integrates with Pandas DataFrames, making it convenient for data analysis tasks.

4. Specialized Plot Types: It offers functions for creating specialized plots like violin plots, box plots, pair plots, and more, which are particularly useful for exploring data distributions and relationships.

Seaborn is commonly used in data analysis, data science, and machine learning projects for its ability to quickly generate informative and aesthetically pleasing visualizations.

Code and Outputs:

First we are required to import all the necessary packages and import them as libraries.

```
%matplotlib inline
```

Here we have imported the matplot libraries for the data visualization needed for the given problem.

```
Import pandas as pd
```

Here we have imported pandas libraries as pd

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Seaborn is a library for making statistical graphis in phyton.it
Builds on top of matplotlib and integrates closely with pandas
Data stuctures.seaborn helps you explore and understand your data.

```
#plt.style.use('ggplot')
```

For reading the data set in csv file we use the syntax pd.read_csv()for treading the dataset.

df = pd.read_csv("PJME_hourly.csv")

For reading the data set in csv file we use the syntax pd.read_csv()for treading the dataset.

import numpy as np

import pandas as pd

import seaborn as sns

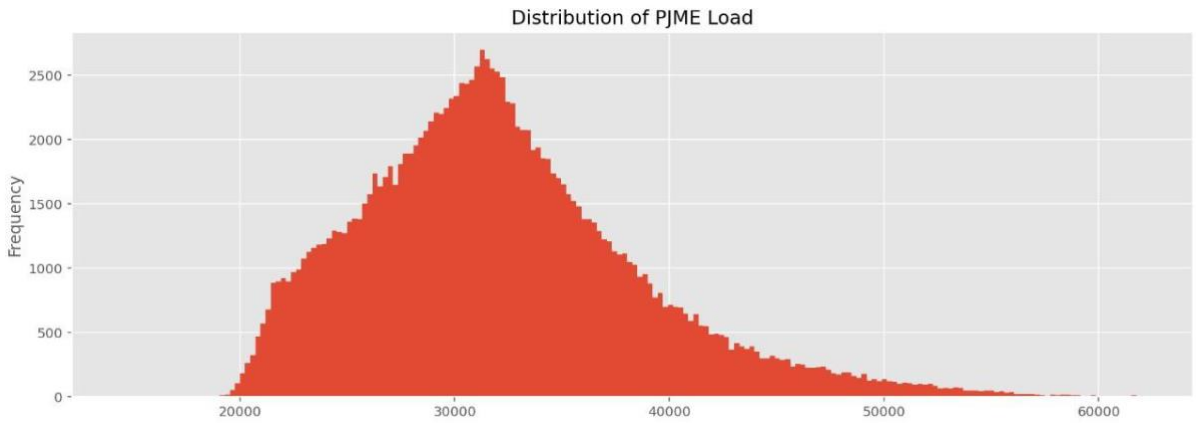
import matplotlib.pyplot as plt

plt.style.use('ggplot')

df = pd.read_csv("PJME_hourly.csv")

f1=data['PJME_MW'].plot.hist(figsize=(15, 5), bins=200, title='Distribution of PJME Load')

OUTPUT:



Histogram plotting is a common data visualization technique used to display the distribution of a dataset. It provides a way to represent the frequency or count of data points within predefined intervals, or "bins," along a continuous or discrete variable. Here's how to create a histogram plot:

1. Data Collection: Gather the dataset you want to visualize. This could be a list of values, a column from a DataFrame, or any suitable data source.

2. Choose the Number of Bins: Decide on the number of bins (intervals) you want to divide your data into. This choice can impact the appearance and interpretability of the histogram.

3. Plotting: In Python, you can use libraries like Matplotlib or Seaborn to create a histogram plot.

4. Interpretation: The resulting histogram will show the distribution of your data, with the x-axis representing the variable's values and the y-axis showing the frequency or count of data points in each bin.

Histograms are particularly useful for visualizing the shape of data distributions, identifying central tendencies, and detecting outliers. They are commonly used in data analysis, statistics, and various scientific and data-driven fields.

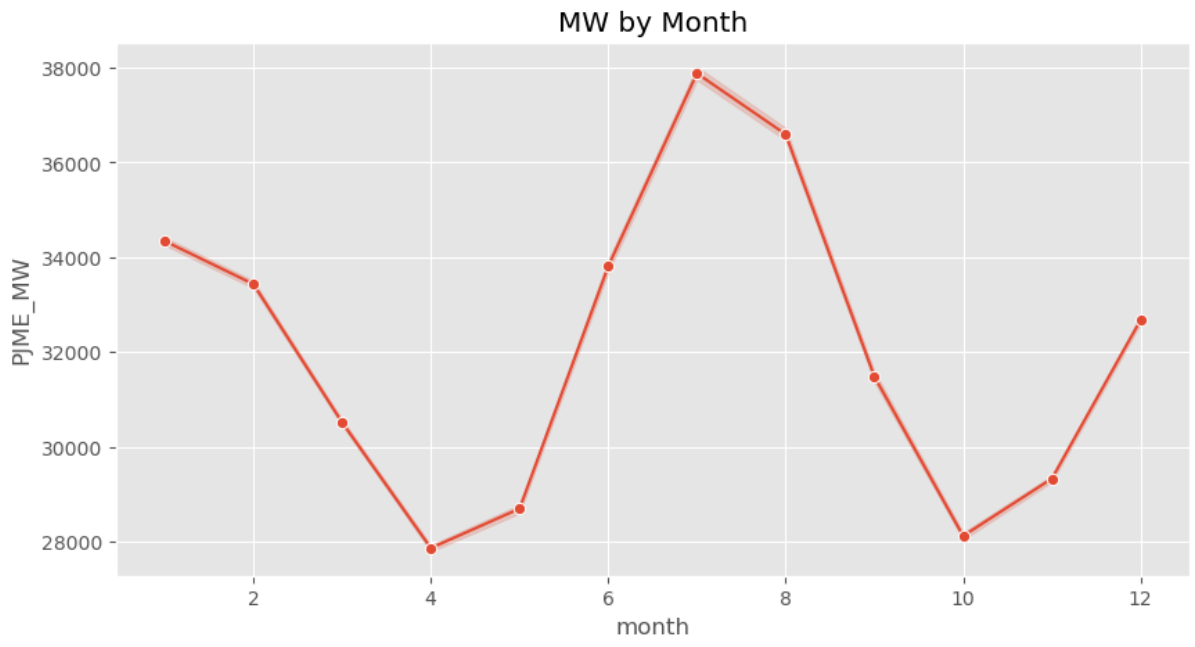
Line Plot:

Displays data points as a series of connected line segments. It's useful for showing trends over time.

Here is the code for line plot:

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.lineplot(data=df, x='month', y='PJME_MW', palette='Blues',marker='o')
ax.set_title('MW by Month')
plt.show()
```

OUTPUT:



Box Plotting:

Creating a box plot (box-and-whisker plot) in Python is straightforward using libraries like Matplotlib or Seaborn. Box plots are useful for visualizing the distribution and spread of data, including outliers.

The code used for the box plot is:

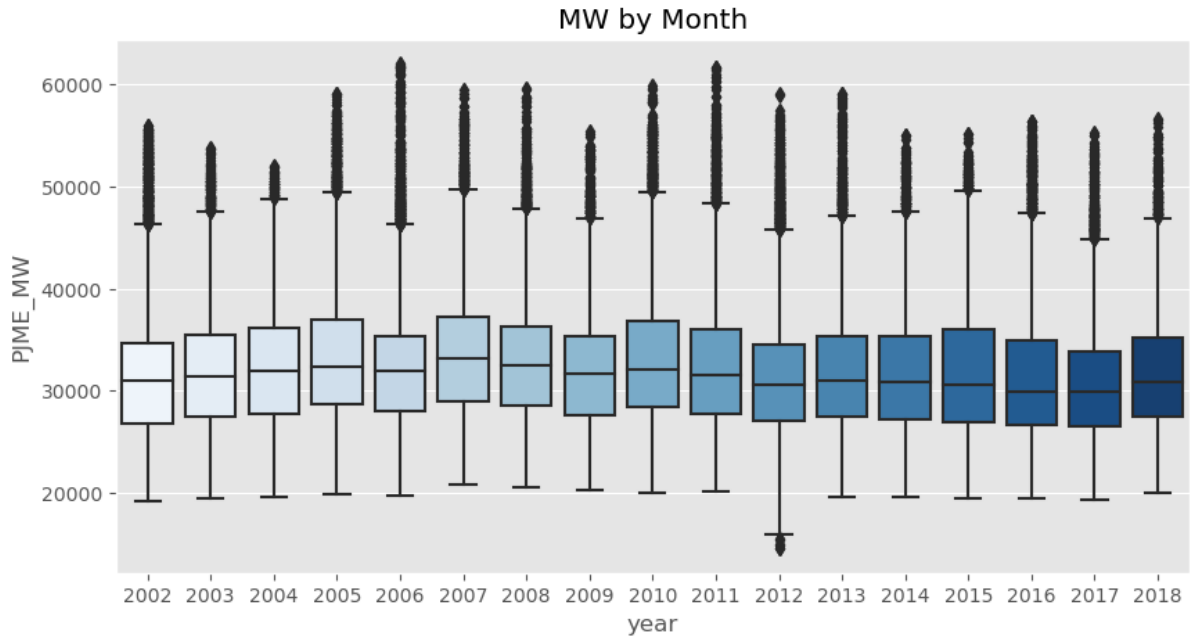
```
plt.boxplot(data)
```

Using Seaborn provides a more visually appealing and informative box plot. In both cases, you can replace the data list with your own dataset for analysis and visualization. Box plots are useful for comparing the distributions of different groups or variables.

The code is given as follows:

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.boxplot(data=df, x='year', y='PJME_MW', palette='Blues')
ax.set_title('MW by Month')
plt.show()
```

OUTPUT:



Prediction :

To make predictions, you typically need a predictive model that has been trained on historical data. Below is a general overview of the process for making predictions:

1. Data Collection: Gather the data you want to make predictions on. This data should be in a format that your predictive model can understand.

2. Data Preprocessing: Preprocess the data to clean it and make it suitable for input into the model. This may include handling missing values, encoding categorical variables, and scaling or normalizing features.

3. Load or Train a Model: Depending on your task, you may need to load a pre-trained model or train a new one. Common libraries for machine learning in Python include scikit-learn, TensorFlow, and PyTorch.

4. Feature Extraction: If your data has a different format or structure from what your model expects, you might need to extract relevant features from the data.

5. Predictions: Use the model to make predictions on your data. The exact code for this step depends on the library and model you're using.

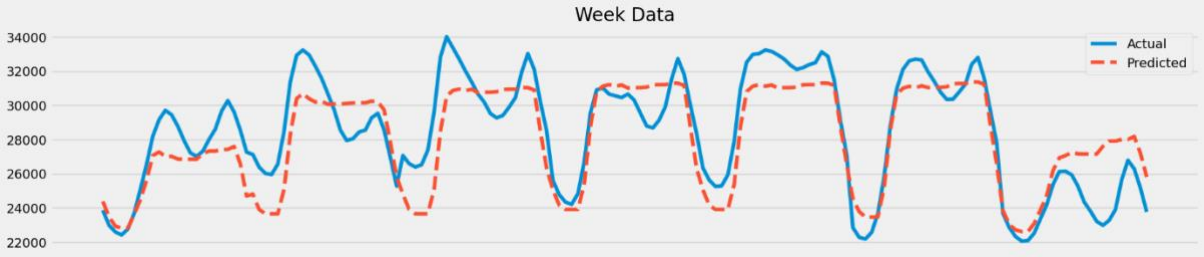
6. Post-processing: After obtaining predictions, you may need to post-process the results, depending on your specific application.

The specific steps and code will depend on your data, the problem you're solving, and the machine learning or predictive modeling library you are using. Be sure to choose the appropriate model and preprocess your data correctly for your specific prediction task.

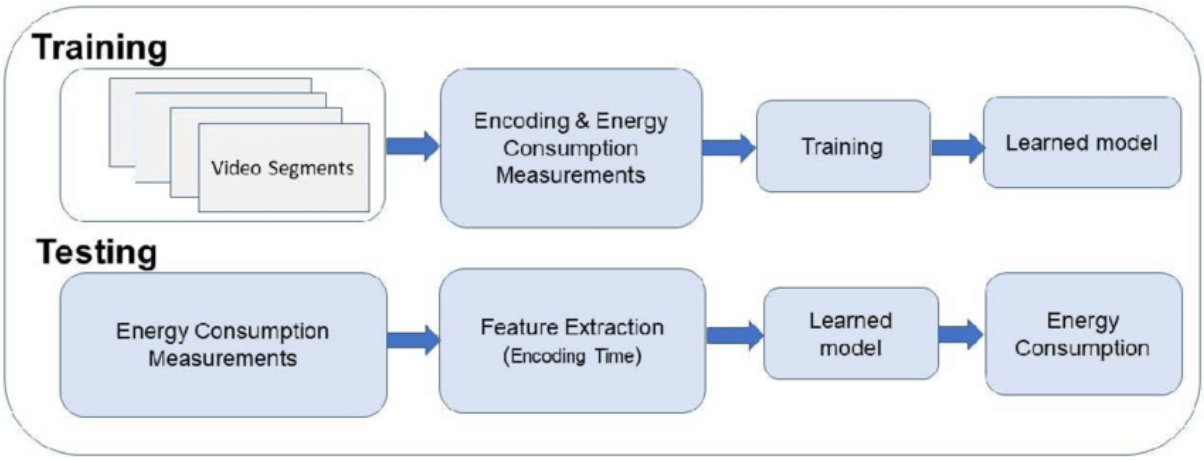
The code for the prediction of the dataset is given by the following codes:

```
ax = df.loc[(df.Datetime > '2018-04-01') & (df.Datetime <= '2018-04-08')]['PJME_MW'].plot(figsize=(20,5), title='Week Data')
df.loc[(df.Datetime > '2018-04-01') & (df.Datetime <= '2018-04-08')]['prediction'].plot(style='--')
plt.legend(['Actual', 'Predicted'])
plt.show()
```

OUTPUT:



ENERGY CONSUMPTION ARCHITECTURE DIAGRAM:



In this project we can divide it into three modules,

These three modules are,

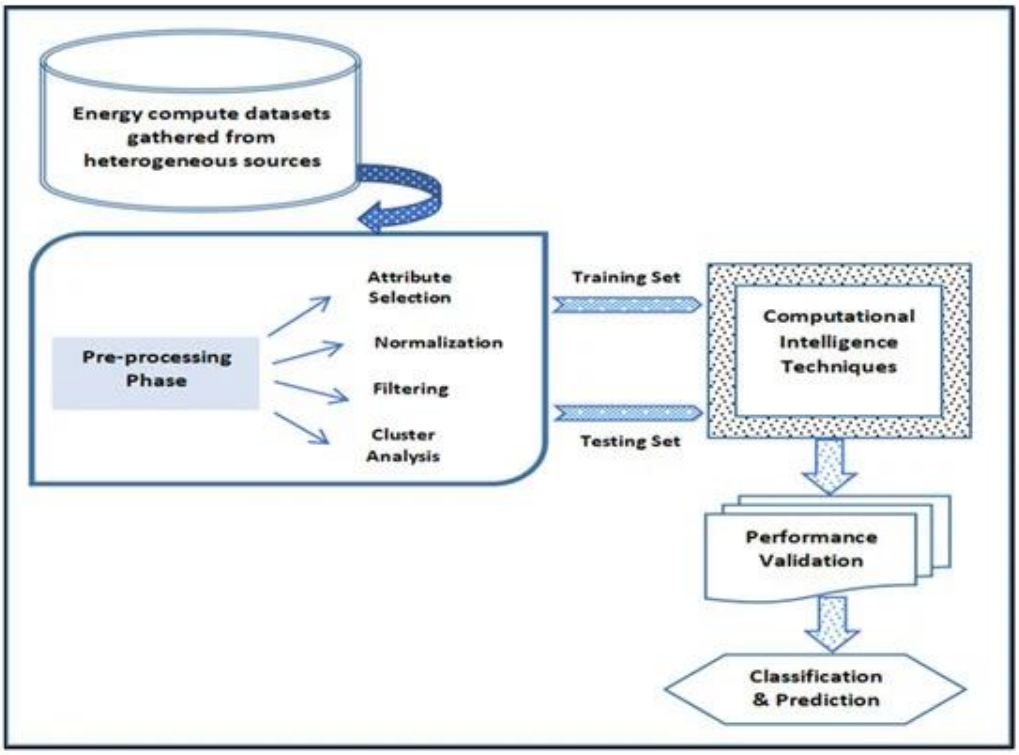
1. Data Preprocessing
2. Algorithm used
3. Data Visualization

Data Preprocessing:

Data preprocessing is a crucial step in preparing your data for analysis or machine learning. It involves cleaning, transforming, and organizing raw data into a suitable format for analysis or model training.

Here are some common data preprocessing tasks:

1. Data Cleaning
2. Data Transformation
3. Data Imputation
4. Data Splitting
5. Data Normalization/Standardization
6. Dealing with Imbalanced Data
7. Text Data Preprocessing
8. Date and Time Data Handling
9. Handling Skewed Data
10. Handling High Cardinality Data



Data preprocessing can significantly impact the quality of your analysis and the performance of machine learning models. The specific steps you need to perform depend on the nature of your data and the objectives of your analysis or modeling task. It's often an iterative process, and domain knowledge plays a crucial role in making informed decisions during data preprocessing

Algorithm used:

The choice of algorithm in data analysis or machine learning depends on the specific task and the characteristics of your dataset. Here are some commonly used algorithms for various types of tasks,

1. Classification
2. Regression
3. Clustering
4. Dimensionality Reduction
5. Natural Language Processing (NLP)
6. Time Series Analysis
7. Deep Learning

The choice of algorithm also depends on factors like the size of your dataset, the amount of available data, computational resources, and the specific goals of your project. It often involves experimenting with different algorithms and evaluating their performance to select the one that works best for your particular use case.

Data Visualization:

Data Visualization is a technique of presenting data graphically or in a pictorial format which helps to understand large quantities of data very easily. This allows decision-makers to make better decisions and also allows identifying new trends, patterns in a more efficient way.