Measure Energy Consumption

NAME:KARUPPAYI.C
REGISTER NO:720421104026

AI_PHASE 2 DOCUMENT SUBMISSION

PROJECT : Measure Energy Consumption

PROBLEM DEFINITION :

The problem at hand is to create and automated system that measure energy consumption,analyses the data, and provides visualization for informed decision-making.This solution aims to enhance efficiency,accuracy,and ease of understanding in managing energy consumption across various sectors.

DESIGN THINKING :

1.Data source: Identify an available dataset containing energy consumption measurements.

2.Data preprocessing: Clean,transform,and prepare the dataset for analysis.

3.Feature Extraction: Extract relevant features and metrics from the energy consumption data.

4.Model Development: Utilize statistical analysis to uncover trends,patterns,and anomalies in the data.

5.Visualization: Develop visualizations (graphs,charts)to present the energy consumption trends and insights.

6. Automation; Build a script that automates data collection,analysis,and visualization processes.

EXPLANATION:

1. *Data Collection*:

- Implement sensors or devices to collect energy consumption data. These could be smart meters, IoT devices, or other sensors connected to the energy sources.

2. *Data Storage*:

- Set up a database system to store the collected data. You can use databases like MySQL, PostgreSQL, or NoSQL databases like MongoDB, depending on the volume and nature of the data.

3. *Data Analysis*:

- Develop algorithms and scripts to analyze the energy consumption data. You can use programming languages like Python with libraries like pandas and NumPy for data analysis.

4. *Visualization*:

- Create a web-based or desktop application to visualize the analyzed data. You can use libraries like Matplotlib, Plotly, or JavaScript frameworks like D3.js for interactive visualizations.

5. *User Interface*:

- Build a user-friendly interface for users to interact with the system. This could be a web application using frameworks like React, Angular, or Vue.js, or a desktop application using technologies like PyQt or Electron.

6. *Automation*:

- Implement automation scripts to schedule data collection, analysis, and reporting tasks.

7. *Security*:

- Ensure data security and access control mechanisms to protect sensitive energy consumption data.

8. *Reporting*:

- Generate reports and alerts based on energy consumption trends and anomalies.

9. *Scaling*:

- Consider scalability as your system may need to handle a large amount of data as it grows.

10. *Testing and Deployment*:

- Thoroughly test your system before deploying it in a production environment.

11. *Maintenance and Updates*:

- Regularly maintain and update your system to ensure it remains accurate and secure.

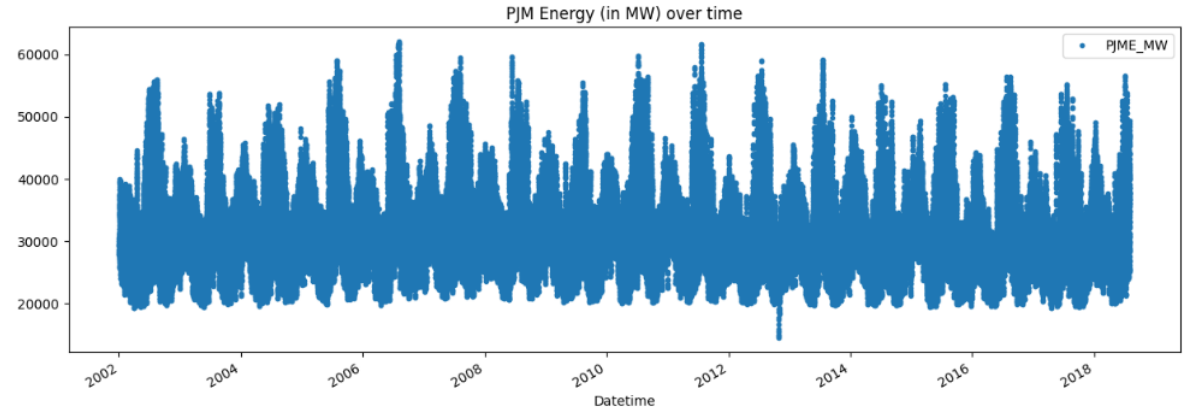
DATA SOURCE:

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

Program:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


# customize the style

pd.options.display.float_format = '{:.5f}'.format

pd.options.display.max_rows = 12


# load the data

filepath = '../input/hourly-energy-consumption/PJME_hourly.csv'

df = pd.read_csv(filepath)


print("Now, you're ready for step one")


# 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()



 # 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()

 # 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)

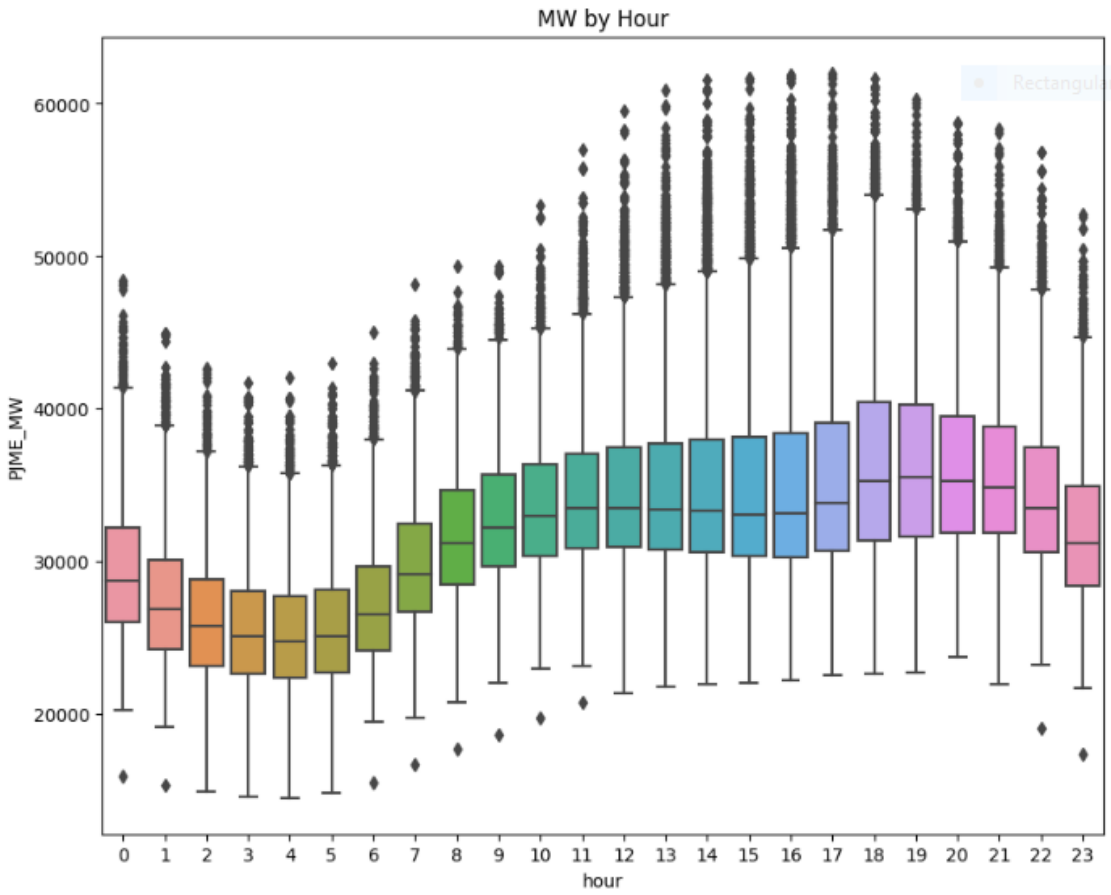
 # visualize the hourly Megawatt

fig, ax = plt.subplots(figsize=(10, 8))

sns.boxplot(data=df, x='hour', y='PJME_MW')

ax.set_title('MW by Hour')

plt.show()



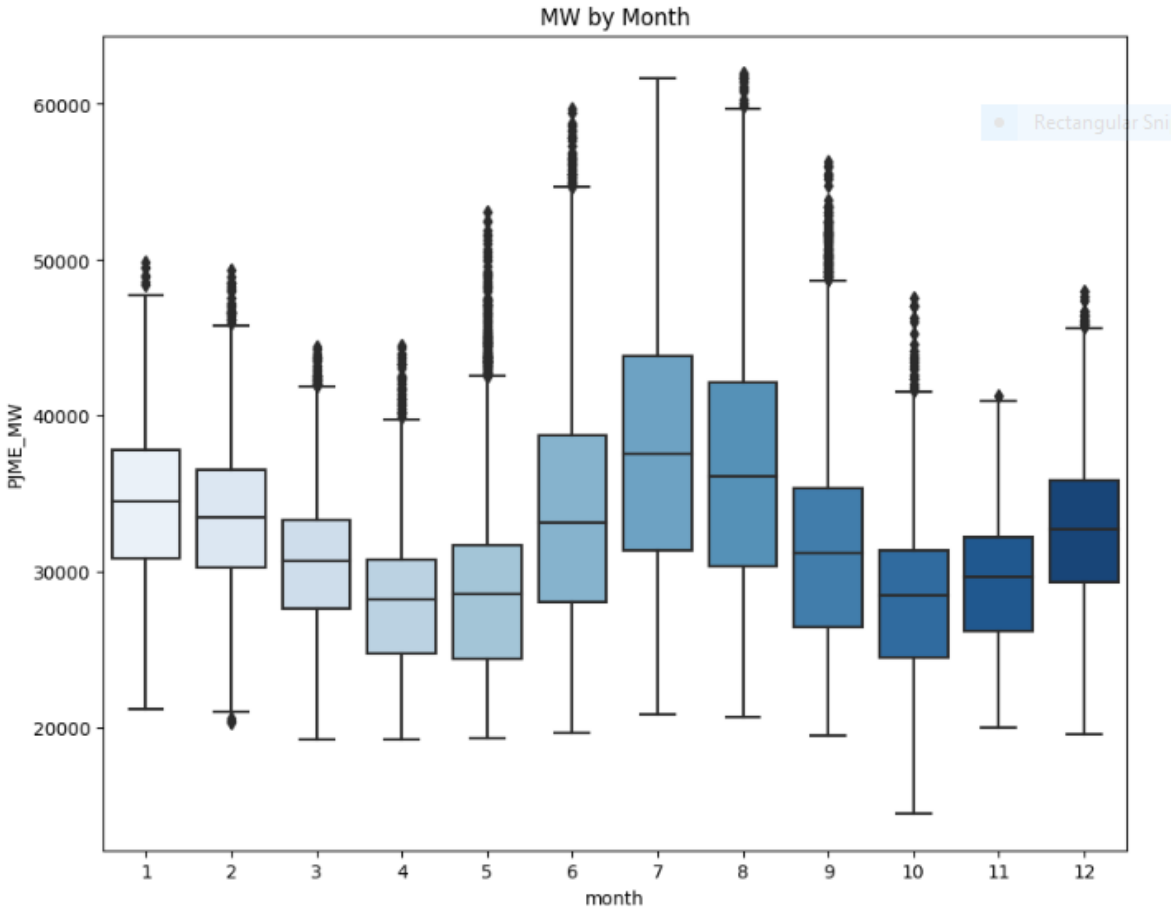
 # viaualize the monthly Megawatt

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()



 # preprocessing

train = create_features(train)

test = create_features(test)

features = ['dayofyear', 'hour', 'dayofweek', 'quarter', 'month', 'year']

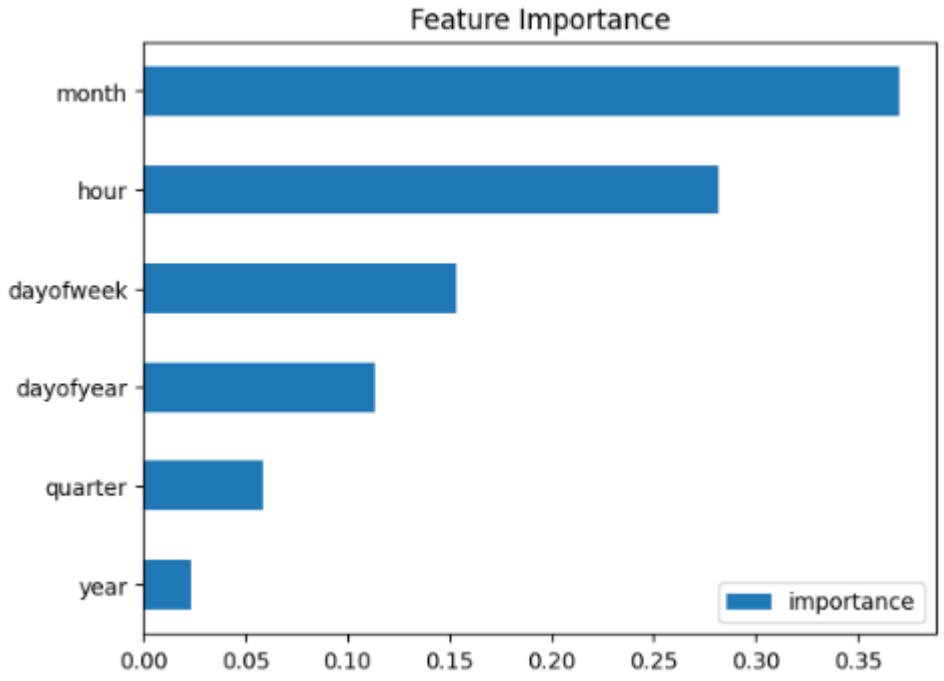
target = 'PJME_MW'

X_train = train[features]

y_train = train[target]

```
X_test = test[features]

y_test = test[target]


import xgboost as xgb

from sklearn.metrics import mean_squared_error


# build the regression model

reg = xgb.XGBRegressor(base_score=0.5, booster='gbtree',

                       n_estimators=1000,

                       early_stopping_rounds=50,

                       objective='reg:linear',

                       max_depth=3,

                       learning_rate=0.01)

reg.fit(X_train, y_train,

        eval_set=[(X_train, y_train), (X_test, y_test)],

        verbose=100)


fi = pd.DataFrame(data=reg.feature_importances_,

             index=reg.feature_names_in_,

             columns=['importance'])

fi.sort_values('importance').plot(kind='barh', title='Feature
Importance')
```

plt.show()



---

test['prediction'] = reg.predict(X_test)

df = df.merge(test[['prediction']], how='left', left_index=True, right_index=True)

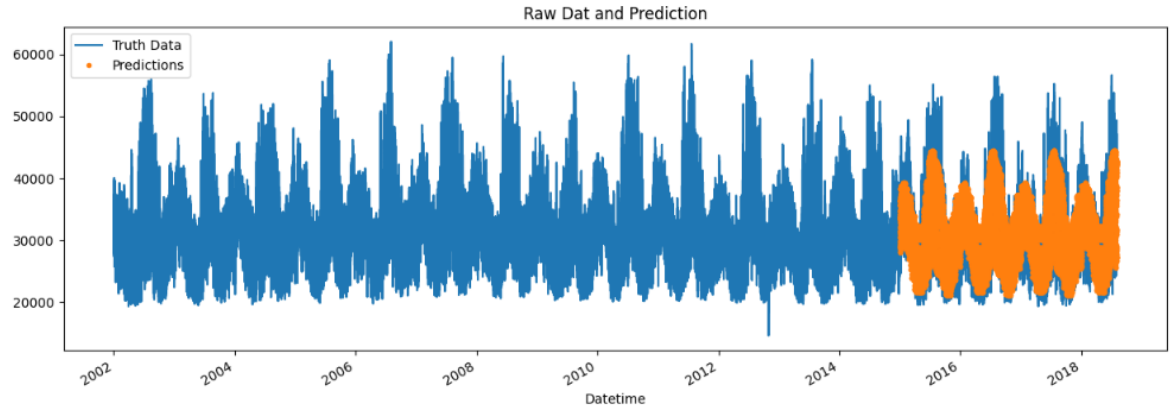
ax = df[['PJME_MW']].plot(figsize=(15, 5))

df['prediction'].plot(ax=ax, style='.')

plt.legend(['Truth Data', 'Predictions'])

ax.set_title('Raw Dat and Prediction')

plt.show()



CONCLUSION:

In conclusion, the development of an automated energy consumption monitoring system with data analysis and visualization is a complex and multifaceted task. It encompasses various stages, including data collection, storage, analysis, visualization, and user interface development. Ensuring data security, scalability, and reliable automation further add to the complexity. Regular maintenance and updates are necessary to keep the system accurate and secure. This endeavor demands a multidisciplinary team with expertise in hardware, software, data science, and user interface design.