**Measure energy consumption**

**Vasanth Ram P**

**au410721104121**

**Phase-4: Development Part 2**

**Project: Measure Energy Consumption**

**Phase-4: Development Part 2**

**Topic: In this part you will continue building your project.**

**Continue the development by:**

**Analyzing the energy consumption data**

**Creating visualizations.**

**Email: ganeshkumarreddyt.cse2021@dce.edu.in**

**Analyzing the energy consumption data:**

Analyzing energy consumption data involves several key steps, including data collection, data exploration, visualization, and data loading and preprocessing. Here's a breakdown of these steps in the context of analyzing energy consumption data:

**Data Collection:**

* Gather energy consumption data from various sources, such as utility companies, smart meters, or sensors. Ensure that the data is complete and accurate.
* Organize the data into a structured format, which may include columns for date and time, energy consumption values, and any relevant contextual information (e.g., temperature, occupancy).

**Data Exploration:**

* Begin by performing basic data exploration to understand the structure and characteristics of the dataset.
* Check for missing values, outliers, and data quality issues. Address any anomalies to ensure data integrity.
* Examine summary statistics (e.g., mean, median, standard deviation) to get an initial sense of the data's distribution.

**Data Loading and Preprocessing:**

* Load the energy consumption data into a data analysis tool or programming environment such as Python, R, or Excel.
* Perform preprocessing tasks to clean and format the data for analysis:
  + - Handle missing data: Fill in missing values using interpolation or other appropriate methods.
    - Remove outliers: Identify and remove extreme values that could distort the analysis.
    - Standardize units: Ensure that all data is in consistent units of measurement.
    - Transform data: If necessary, apply transformations (e.g., logarithmic transformation) to make the data more suitable for analysis.
* Organize the data into a structured dataset with well-defined columns, such as Date, Energy Consumption, and any additional relevant variables.

**Visualization:**

* Create visualizations to gain insights into energy consumption patterns. Common energy consumption visualizations include:
  + - Line charts: Displaying energy consumption over time to identify trends and seasonality.
    - Histograms: Illustrating the distribution of energy consumption values.
    - Box plots: Revealing statistical measures like quartiles and outliers.
    - Scatter plots: Exploring relationships between energy consumption and other variables (e.g., temperature).
* Use color coding, labels, and legends to make your visualizations more informative and understandable.

**Energy Consumption Patterns:**

* Identify daily, weekly, monthly, and yearly patterns in energy consumption.
* Look for anomalies or unusual spikes in consumption that may need further investigation.

**Evaluating the performance:**

* Evaluating the performance of your energy consumption model is crucial to ensure it provides accurate predictions and valuable insights.
* The choice of performance metrics will depend on the specific goals of your analysis and the characteristics of your data.

**Python program:**

**Importing libraries:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import xgboost as xgb

plt.style.use('ggplot')

**Importing dataset**

df=pd.read\_csv('../input/hourly-energy-consumption/PJME\_hourly.csv')

print(df.head(2))

print('--'\*10)

print(df.tail(2))

Datetime PJME\_MW

0 2002-12-31 01:00:00 26498.0

1 2002-12-31 02:00:00 25147.0--

--------------------

Datetime PJME\_MW

145364 2018-01-01 23:00:00 40164.0

145365 2018-01-02 00:00:00 38608.0

df.dtypes

Datetime object

PJME\_MW float64

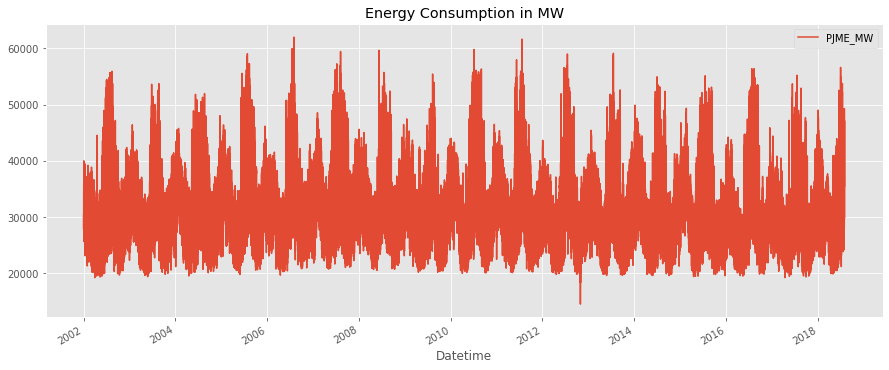
dtype: object

## Visualize the Energy consumption

df.plot(figsize=(15,6),title='Energy Consumption in MW')

o/p-

<AxesSubplot:title={'center':'Energy Consumption in MW'}, xlabel='Datetime'>



**Train test split**

*# Let's take training data till Dec-2014 and test data since then*

train=df.loc[df.index<'2015-01-01']

test=df.loc[df.index>='2015-01-01']

fig,ax=plt.subplots(figsize=(15,6))

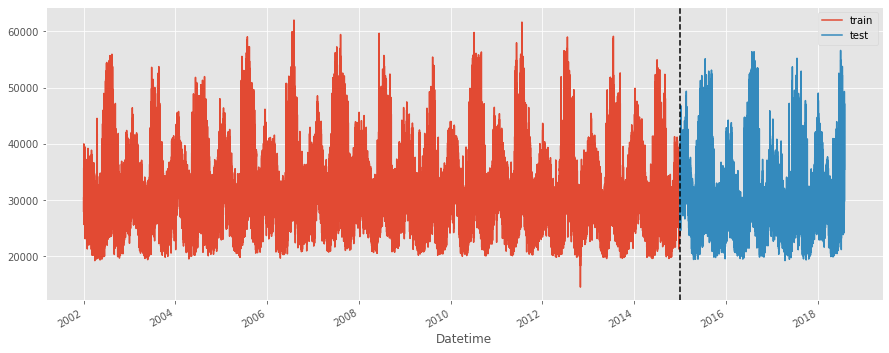
train.plot(ax=ax)

test.plot(ax=ax)

ax.legend(['train','test'])

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

plt.show()

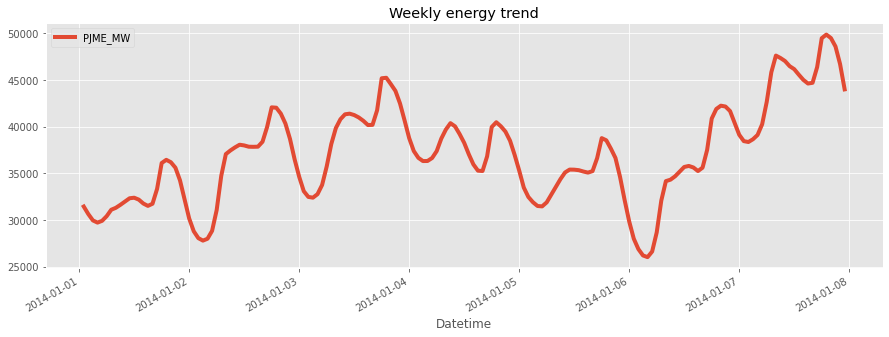


## Visualize 1 week of Data

df.loc[(df.index>'2014-01-01') & (df.index<'2014-01-08')].plot(figsize=(15,5),title='Weekly energy trend',grid='on',lw=4)

o/p-

<AxesSubplot:title={'center':'Weekly energy trend'}, xlabel='Datetime'>

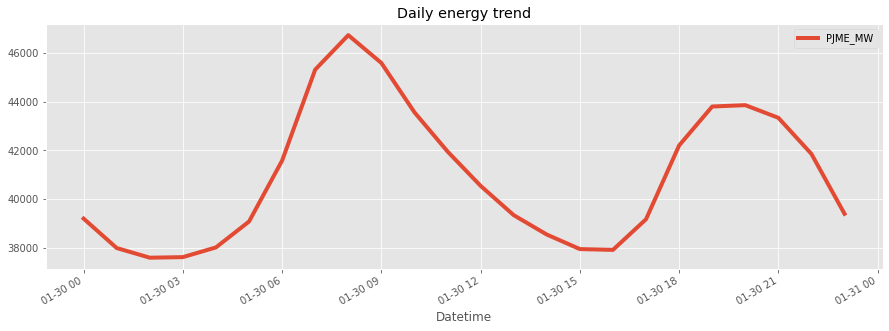


## Visualize daily energy consumption

df.loc[(df.index>='2014-01-30 00:00:00') & (df.index<'2014-01-31 00:00:00')].plot(figsize=(15,5),title='Daily energy trend',grid='on',lw=4)

o/p-

<AxesSubplot:title={'center':'Daily energy trend'}, xlabel='Datetime'>



# Feature Engineering

def get\_feats(df):

df['hour']=df.index.hour

df['dayofweek']=df.index.dayofweek

df['month']=df.index.month

df['quarter']=df.index.quarter

df['year']=df.index.year

df['dayofyear']=df.index.dayofyear

df['weekofyear']=df.index.weekofyear

return df

df=get\_feats(df)

df.head(2)

| Datetime | PJME\_MW | hour | dayofweek | month | quarter | year | Dayof  year | Weekof  year |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2002-12-31 01:00:00 | 26498.0 | 1 | 1 | 12 | 4 | 2002 | 365 | 1 |
| 2002-12-31 02:00:00 | 25147.0 | 2 | 1 | 12 | 4 | 2002 | 365 | 1 |

## Visualize hourly Energy Distribution

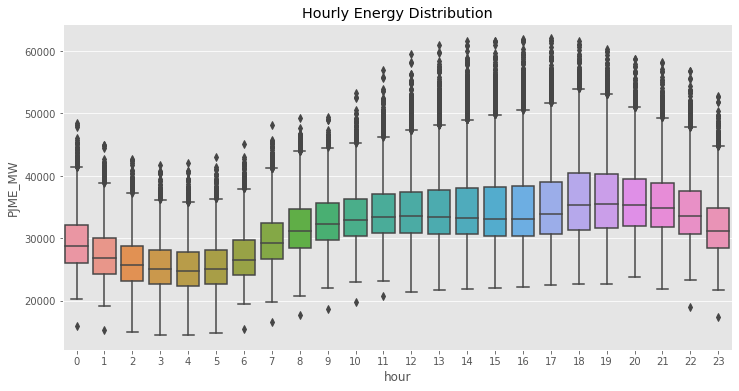
lt.figure(figsize=(12,6))

sns.boxplot(data=df,x='hour',y='PJME\_MW')

plt.title('Hourly Energy Distribution')

o/p-

Text(0.5, 1.0, 'Hourly Energy Distribution')



## Visualize Monthly Energy Distribution

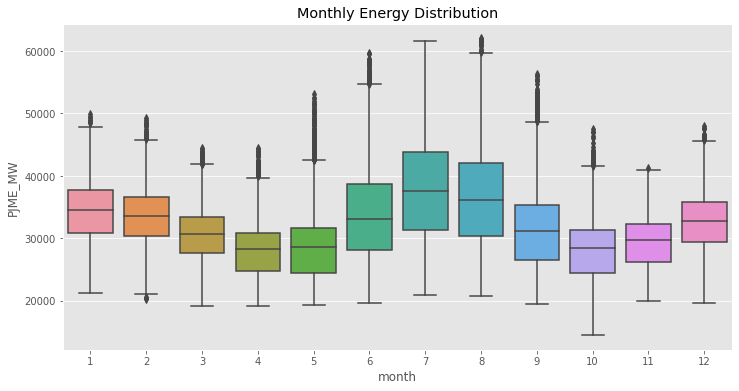
plt.figure(figsize=(12,6))

sns.boxplot(data=df,x='month',y='PJME\_MW')

plt.title('Monthly Energy Distribution')

O/p-

Text(0.5, 1.0, 'Monthly Energy Distribution')



## Visualize Quarterly Energy Distribution

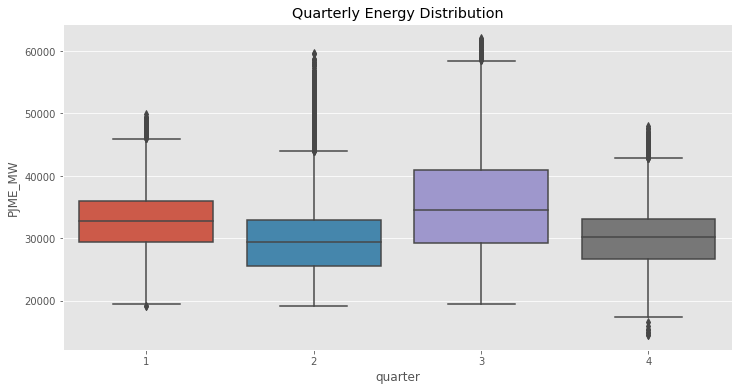
plt.figure(figsize=(12,6))

sns.boxplot(data=df,x='quarter',y='PJME\_MW')

plt.title('Quarterly Energy Distribution')

O/p-

Text(0.5, 1.0, 'Quarterly Energy Distribution')



# Training the XGBoost model

regressor.fit(X\_train,y\_train,eval\_set=[(X\_train,y\_train),(X\_test,y\_test)],verbose=50)

[0] validation\_0-rmse:32765.33654 validation\_1-rmse:31817.63654

[50] validation\_0-rmse:25637.33586 validation\_1-rmse:24700.52578

[100] validation\_0-rmse:20108.70500 validation\_1-rmse:19190.00580

[150] validation\_0-rmse:15835.18985 validation\_1-rmse:14941.06871

[200] validation\_0-rmse:12551.64082 validation\_1-rmse:11701.42494

[250] validation\_0-rmse:10036.88215 validation\_1-rmse:9246.50244

[300] validation\_0-rmse:8132.28239 validation\_1-rmse:7424.02857

[350] validation\_0-rmse:6704.30769 validation\_1-rmse:6105.68263

[400] validation\_0-rmse:5651.16136 validation\_1-rmse:5193.34251

[450] validation\_0-rmse:4887.79011 validation\_1-rmse:4583.75246

[500] validation\_0-rmse:4349.08523 validation\_1-rmse:4205.89242

[550] validation\_0-rmse:3972.19071 validation\_1-rmse:3969.49644

[600] validation\_0-rmse:3711.37849 validation\_1-rmse:3837.76851

[650] validation\_0-rmse:3525.74243 validation\_1-rmse:3764.26414

[700] validation\_0-rmse:3392.55546 validation\_1-rmse:3725.45955

[750] validation\_0-rmse:3297.57502 validation\_1-rmse:3702.92489

[800] validation\_0-rmse:3224.36090 validation\_1-rmse:3693.31029

[850] validation\_0-rmse:3168.04419 validation\_1-rmse:3694.09309

[851] validation\_0-rmse:3166.70494 validation\_1-rmse:3694.28594

O/p--

XGBRegressor(base\_score=0.5, booster='gbtree', callbacks=None,

colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=1,

early\_stopping\_rounds=50, enable\_categorical=False,

eval\_metric =None, gamma=0, gpu\_id=-1, grow\_policy='depthwise',

importance\_type=None, interaction\_constraints='',

learning\_rate=0.005, max\_bin=256, max\_cat\_to\_onehot=4,

max\_delta\_step=0, max\_depth=4, max\_leaves=0, min\_child\_weight=1,

missing=nan, monotone\_constraints='()', n\_estimators=1000,

n\_jobs=0, num\_parallel\_tree=1, predictor='auto', random\_state=0,

reg\_alpha=0, reg\_lambda=1, ...)

# Feature Importance

feat\_imp=pd.DataFrame(data=regressor.feature\_importances\_,index=regressor.feature\_names\_in\_,columns=['importance'])

feat\_imp

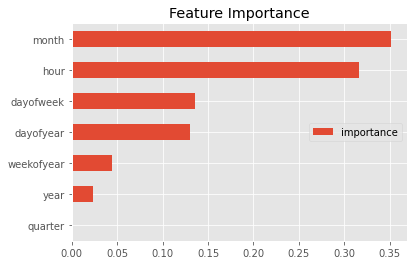
O/p--

|  |  |
| --- | --- |
|  | Importane |
| hour | 0.316475 |
| dayofweek | 0.135843 |
| month | 0.350815 |
| quarter | 0.000000 |
| year | 0.023166 |
| dayofyear | 0.129898 |
| weekofyear | 0.043803 |

feat\_imp.sort\_values('importance').plot(kind='barh',title='Feature Importance').legend(loc='right')

O/p--

<matplotlib.legend.Legend at 0x7ff9fed8a650>



# Visualize our prediction

forecast\_df=df.loc[df.index>='2015-01-01']

forecast\_df.sample(3)

O/p--

| Datetime | PJME\_MW | hour | Dayof  week | month | quarter | year | Dayof  year | Weekof  year | forecast |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2017-04-17 08:00:00 | 26830.0 | 8 | 0 | 4 | 2 | 2017 | 107 | 16 | 29983.414062 |
| 2018-07-11 20:00:00 | 44899.0 | 20 | 2 | 7 | 3 | 2018 | 192 | 28 | 42399.035156 |
| 2015-06-08 07:00:00 | 26291.0 | 7 | 0 | 6 | 2 | 2015 | 159 | 24 | 27769.464844 |

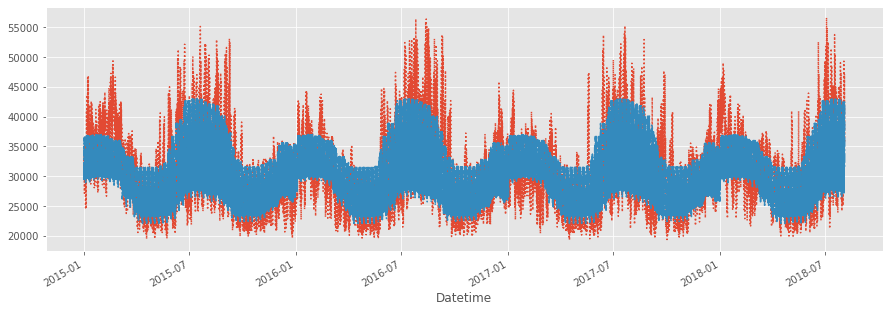
fig,ax=plt.subplots(figsize=(15,5))

forecast\_df['PJME\_MW'].plot(ax=ax,ls=':')

forecast\_df['forecast'].plot(ax=ax,ls='-.')

O/p--

<AxesSubplot:xlabel='Datetime'>



fig,ax=plt.subplots(figsize=(15,5))

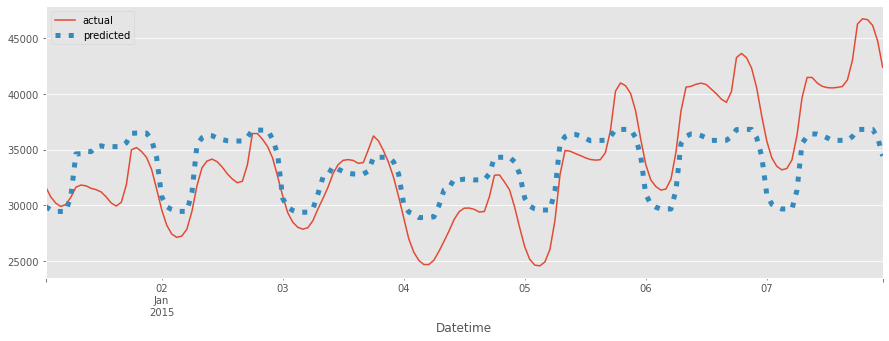
forecast\_df.loc[(forecast\_df.index>'2015-01-01') & (forecast\_df.index<'2015-01-08')]['PJME\_MW'].plot(ax=ax,ls='-')

forecast\_df.loc[(forecast\_df.index>'2015-01-01') & (forecast\_df.index<'2015-01-08')]['forecast'].plot(ax=ax,ls=':',lw=5)

ax.legend(['actual','predicted'])

O/p--

<matplotlib.legend.Legend at 0x7ffa05324d10>



# Evaluating our Model (or) performance

from sklearn.metrics import mean\_squared\_error

rmse=np.sqrt(mean\_squared\_error(forecast\_df['PJME\_MW'],forecast\_df['forecast']))

print('RMSE on Test set is',rmse)

RMSE on Test set is 3692.9683324341677