**MEASURE ENERGY CONSUMPTION**

**NAME: AAFRIN NISHA.M REG NO: 110521104001**

**PROJECT TITLE: MEASURE ENEGRY CONSUMPTION**

**PHASE 04: Development Part2**

**TOPIC: To measure energy consumption by performing different activities like feature engineering, model training, evaluation etc.**



**MACHINE LEARNING OPERATIONS**

Modelling &

Evaluation

Data Engineering

|  |  |  |  |
| --- | --- | --- | --- |
| Distribution Transformations | Feature Importance | Synthetic Data | Advanced Techniques |

# INTRODUCTION:

Feature

engineering

Measure energy consumption is useful for performing and processing of machine learning operations like data engineering, feature engineering, modelling & evaluation is also essential in the field of data science. In the above flow diagram as shown several operations.

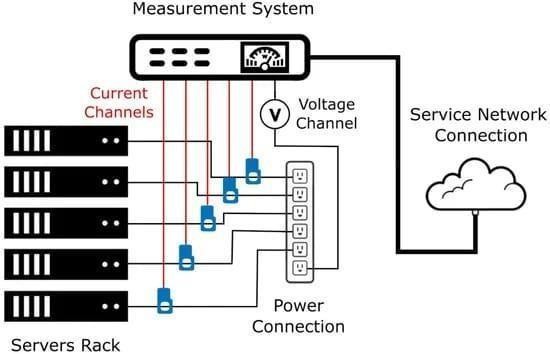
It contains a crucial consumption models and gaining values insights from data.

# Data collection:

The first step is to gather relevant data from various sources, ensuring it's clean and well-structured.

Data collection is used to gathering and storing large datasets can consume energy, especially when dealing with data centers and cloud services. Using efficient data collection techniques can help reduce energy usage.

It also produces a digital transition that drives the new industrial revolution is largely driven by the application of intelligence and data. This boost leads to an increase in energy centers.



The figure shows the data connection architecture diagram. The measurement system is connected to the server’s power input to measure their energy consumption and send the information over a computer networks.

# Necessary step to follow:

1. Import libraries: start by importing the necessary libraries.

## Program:

import pandas as pd import numpy as np

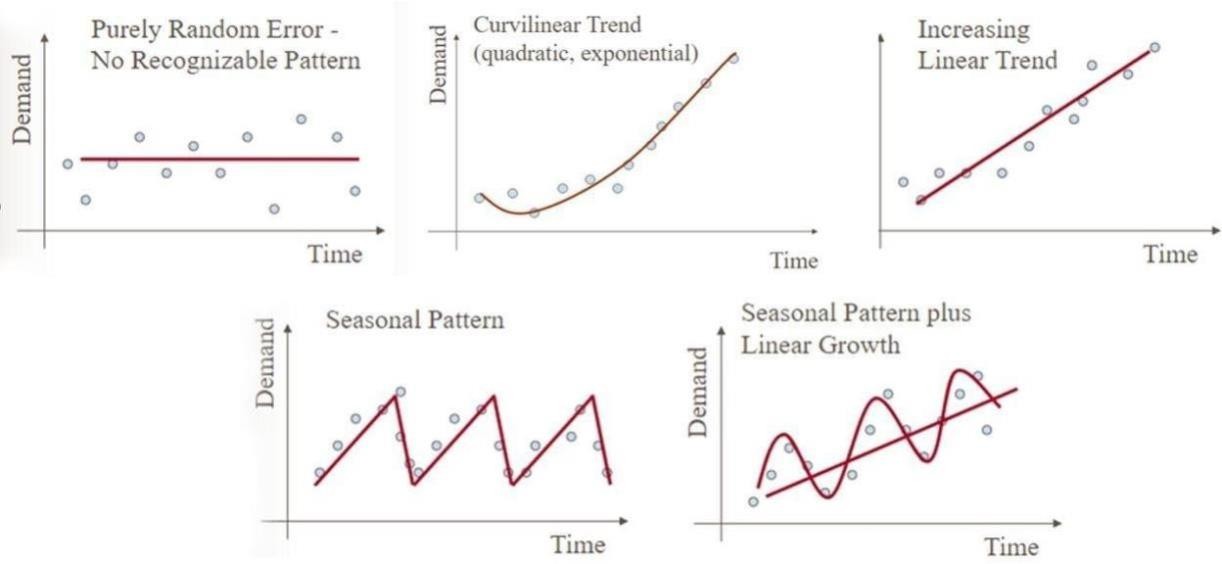
import matplotlib.pyplot as plt import seaborn as sns

mport pandas as pd import xgboost as xgb

Time series data by using mean\_square\_red\_error:

import xgboost as xgb

from sklearn.metrics import mean\_squared\_error color\_pal = sns.color\_palette() plt.style.use('fivethirtyeight')



The figure shows that the different types of time series data forecasting with machine learning.

**Data Processing:**

Data processing involves tasks like data cleaning, handling missing values, and encoding categorical variables to prepare the data for analysis.

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

df = df.set\_index('Datetime') df.index = pd.to\_datetime(df.index) df.plot(style='.',

figsize=(15, 5), color=color\_pal[0],

title='PJME Energy Use in MW') plt.show()

**Dataset for Date Time PJMW MW:**

|  |  |  |
| --- | --- | --- |
| 0 | 2002-12-31  01:00:00 | 5077.0 |
| 1 | 2002-12-31  02:00:00 | 4939.0 |
| 2 | 2002-12-31  03:00:00 | 4885.0 |
| 3 | 2002-12-31  04:00:00 | 4857.0 |
| 4 | 2002-12-31  05:00:00 | 4930.0 |

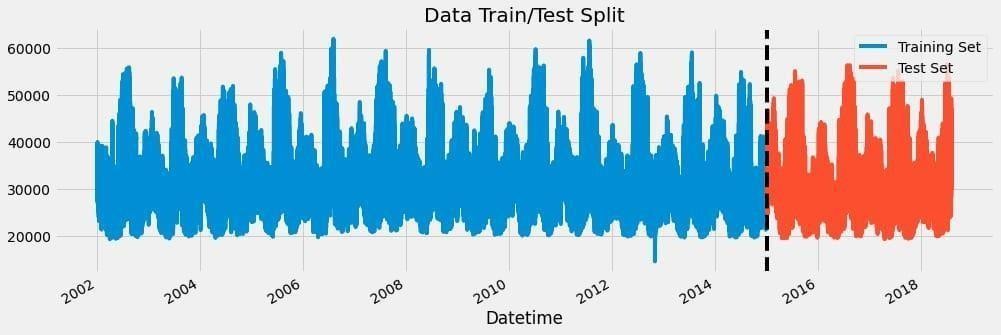
# Training and testing of data split:

Splitting data into training and testing sets is crucial for measuring energy consumption using time series in machine learning. It helps evaluate the model's performance on unseen data.

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

train.plot(ax=ax, label='Training Set', title='Data 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:

It involves creating new features from the existing data to improve the accuracy of the energy consumption prediction.

In order to measure energy consumption and forecast time series data, plays a vital role of feature engineering.

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)

//creation of time series features based on time series index.

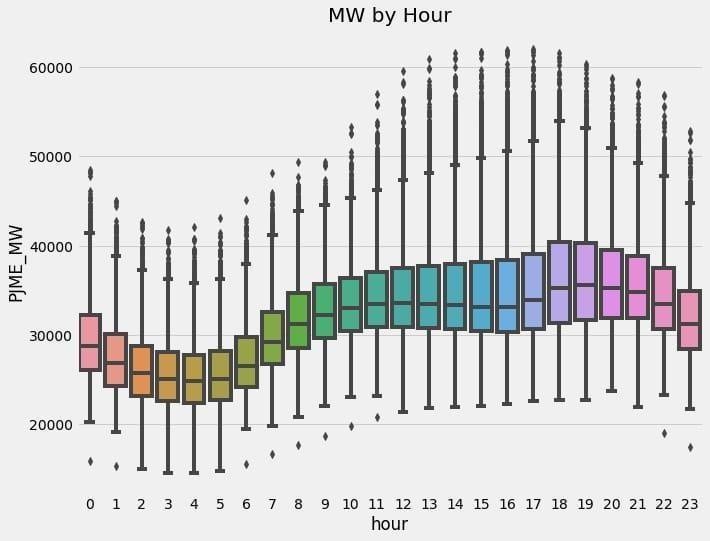
# Visualization of feature and target relationship:

It helps to understand pattern and correlations.

It also allows to identify the features of impact in energy consumption.

fig, ax = plt.subplots(figsize=(10, 8)) sns.boxplot(data=df, x='hour', y='PJME\_MW') ax.set\_title('MW by Hour')

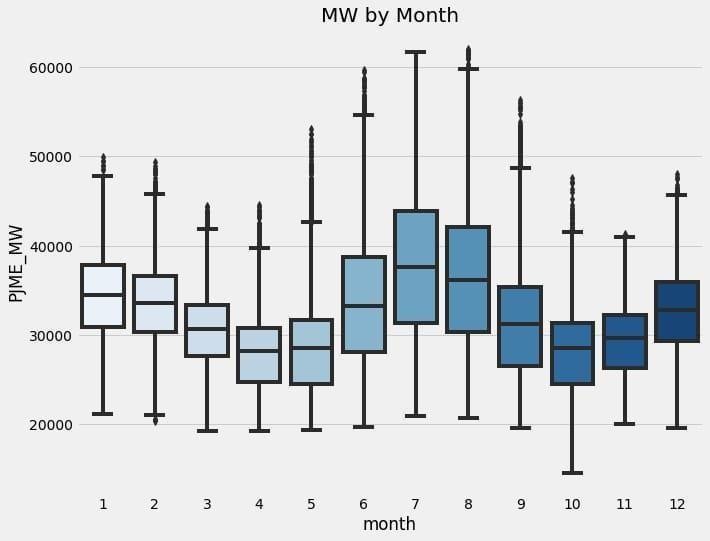
plt.show()



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()



# Creation of models and training:

Creation of models is choosing an appropriate machine learning algorithm that matches the problem at hand, and fine-tuning hyper parameters for optimal performance.

Training the selected model on the preprocessed data to learn patterns and relationships within the data.

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]

## Program by using regression function:

X\_test = test[FEATURES] y\_test = test[TARGET]

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)

## Output:

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=0.01, max\_bin=256, max\_cat\_to\_onehot=4,

max\_delta\_step=0, max\_depth=3, max\_leaves=0, min\_child\_weight=1,

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

n\_jobs=0, num\_parallel\_tree=1, objective='reg:linear', predictor='auto', random\_state=0, reg\_alpha=0, ...)

# Feature importance of model:

Feature importance refers to determining the contribution of each feature in predicting energy consumption.

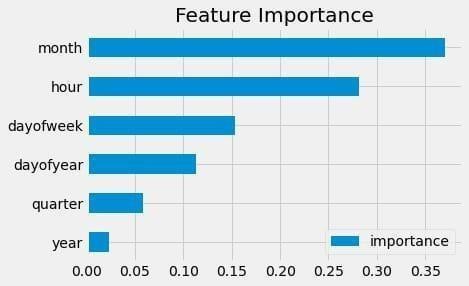
It helps identify which factors have the most influence on energy consumption patterns. This information can guide decision- making and help prioritize actions to optimize energy usage.

Various techniques, such as permutation importance or feature importance from tree-based models, can be used to calculate feature importance.

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()



# Forecasting on data and test prediction:

This allows you to evaluate the model's performance and accuracy in predicting energy consumption.

Forecasting a data is also used for model on historical data and then test its predicting energy consumption.

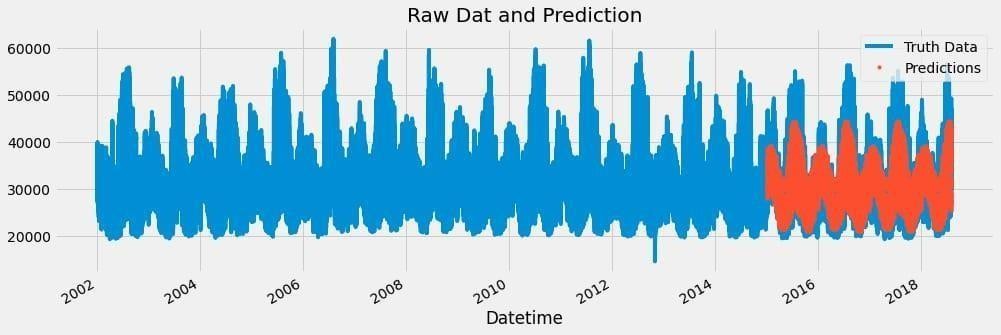
By comparing the model's predictions with the actual energy consumption values, you can assess how well the model is able to forecast future energy consumption patterns.

It's an effective way to validate and fine-tune the machine learning model for accurate energy consumption predictions.

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()



ax = df.loc[(df.index > '04-01-2018') & (df.index < '04-08- 2018')]['PJME\_MW'] \

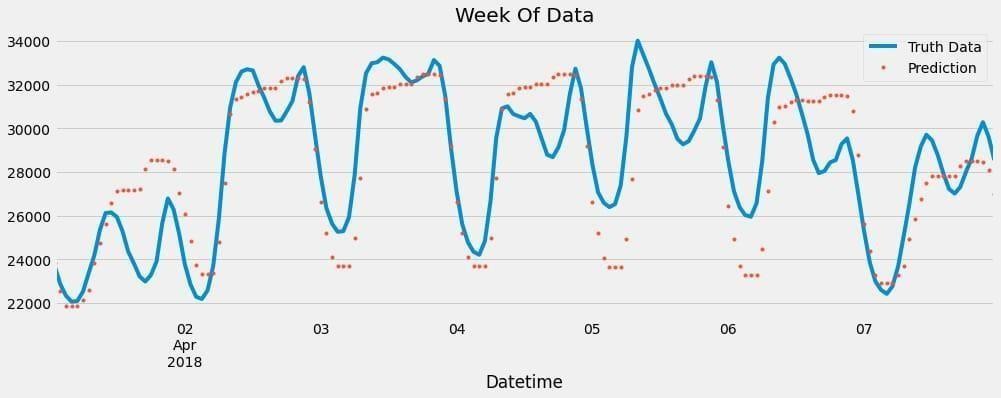
.plot(figsize=(15, 5), title='Week Of Data')

df.loc[(df.index > '04-01-2018') & (df.index < '04-08- 2018')]['prediction'] \

.plot(style='.')

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

plt.show()



# Evaluate and Calculating Errors:

To evaluate and calculate the error in energy consumption prediction,

It performs a various metrics like absolute error (MAE), root mean squared error (RMSE), or mean absolute percentage error (MAPE).

These metrics help quantify the difference between the predicted energy consumption values and the actual values.

By calculating the error, it also assess the accuracy and performance of the prediction model.

test['error'] = np.abs(test[TARGET] - test['prediction'])

test['date'] = test.index.date

test.groupby(['date'])['error'].mean().sort\_values(ascending= False).head(10)

|  |  |
| --- | --- |
| **Output:** |  |
| date |  |
| 2016-08-13 | 12839.595459 |
| 2016-08-14 | 12780.209554 |
| 2016-09-10 | 11356.302002 |
| 2015-02-20 | 10965.976237 |
| 2016-09-09 | 10864.953451 |
| 2018-01-06 | 10506.844889 |
| 2016-08-12 | 10124.050618 |
| 2015-02-21 | 9881.798503 |
| 2015-02-16 | 9781.549805 |
| 2018-01-07 | 9739.143555 |

Name: error, dtype: float64

**Conclusion:**

In conclusion, performing different activities like feature engineering, model training, and evaluation is crucial for accurate energy consumption prediction.

Feature engineering helps in selecting and transforming relevant variables, such as time-based features, weather data, and lagged variables.

Model training involves training the machine learning model on historical data to make accurate predictions.

Evaluation is done by comparing the model's predictions with the actual energy consumption values using metrics like MAE or RMSE.

By following this process, we can build a reliable model that can effectively predict energy consumption patterns.